

FORTRANモデルのラッピング要素モデル作成のサンプル

1. 概要

CommonMPは オブジェクト指向に対応した言語:C#で開発されています。従って、CommonMP上で動作させる要素モデルもC#を用いて開発できるように デバッグ環境等も整備されています。しかしながら、学術計算で用いられるFORTRANを使用される方々にとって、いきなりオブジェクト指向に対応したプログラムを作成するには敷居が高いと感じられるかもしれません。また、既にFORTRANで開発済みのモデルを C#に移植するには、新しい言語の文法を習得するのみならず、オブジェクト指向の思想まで習得する必要があり、その労力から C#に対応した要素モデルを作成することに対して躊躇いが生じるのは無理からぬ事と思われます。そこで、FORTRANで作成したモデルをC#へ移植すること無しに 使用方法として ラッピングが考えられます。ラッピングに関しては、既に CommonMPとしても色々な所で検討がなされており、一部のラッパーがリリースされています。しかしながら、FORTRANプログラムを極力修正せず、汎用的に動作する(どの様なモデルにも対応する) ラッパーを製作することは 困難です。

ここでは、上記ラッパー検討／開発のヒントとして頂くため、簡単なFORTRANモデルをラッピングし、要素モデルとして組み込む方法の一例を取り上げ、サンプルとして示します。

このラッピング用要素モデルは、あくまでも一つの方法を提供するだけで、今後の議論のための参考としてください。

2. FORTRAN ラッピングの方法

ここで示すのは、FORTRANで作成されたモデルをDLL化し、そのDLLをラッパー用要素モデルを介して動作させることで、C#で開発されたCommonMP上で FORTRANで開発されたモデルを動作させる方法です。図1. 1に 本サンプルで説明するFORTRANモデルのラッピング方法の概略を示します。

ここで、FORTRANで開発したモデルを 態々 CommonMP上で動作させる理由としては、

①自らが開発したモデルを別のモデルと組み合わせて使用することで応用範囲を広げたい

②CommonMPの機能を利用してデータベース等の情報を容易に取得し活用したい

③モデルの出力結果をCommonMP-GIS等で表示したい

等が考えられます。このような使用を考えた場合には CommonMP上で単にモデルの計算開始が行えるだけではなく、CommonMPを介して 計算入力や計算結果の出力等の情報の遣り取りが必要となります。そこで、本サンプルでは、モデルを一つのサブルーティンとして動作させ、サブルーティンの引数として、CommonMPとモデル間での情報の交換を行います。

既に開発されたFORTRANモデルに何も手を加えずにCommonMP上で動作出来る事が理想ですが、上記のように CommonMPと情報交換を行うためには モデルの変更も必要となります。そこで、本サンプルでは、ラッピング要素を使用するための FORTRANプログラムのサンプルも合わせて示します。

尚、本サンプルでは、モデルの性能を生かすよりも、汎用的なDLLの組み込み方法の検討に重点が置いてあります。この点、本サンプルを参考とされる方は 独自の工夫をお願いします。

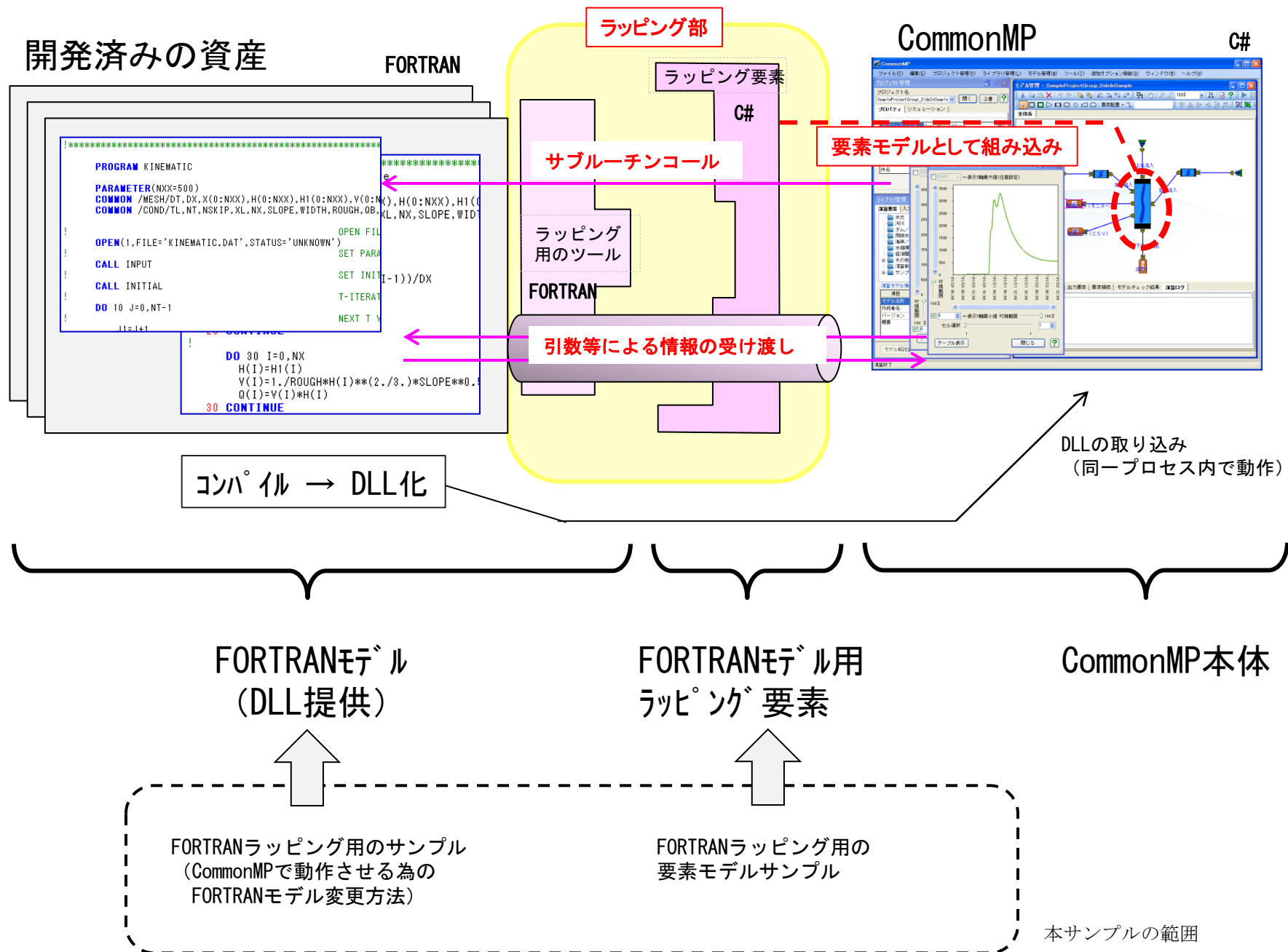


図 2. 1 ラッピング方法概要と 本サンプルの範囲

2. 1 FORTRANモデル開発者の作業概要

図2. 2に FORTRANモデル開発者側が、本サンプルにおけるラッピングを行う場合に意識すべき内容を示します。

まず、モデルをサブルーティンとして作成する必要があります。(既に開発されたFORTRANプログラムが存在するときには、サブルーティン化する必要があります。)

次に、サブルーティンをDLLとしてコンパイルするために必要な宣言等を記述します。これらの宣言は 使用するコンパイラーに依存します。

上記修正を施したソースを元に DLLを作成します。

上記により作成したDLLをCommonMP側で取り込み、ラッピング要素モデルを通じてプロパティ設定や初期値設定等が行えるように、モデル情報を記述したファイル(以後「モデル情報記述ファイル」と呼びます)を作成します。ここで作成したモデル情報記述ファイルとDLLを一对としてCommonMPの所定のディレクトリに置くことで、本サンプルのラッピング要素モデルはFORTRANのモデルを CommonMPの要素モデルの一つとして取り扱うことが可能となります。

(詳細な手順は4章に記述します。)

2. 2 ラッピング要素モデル開発者の作業概要

図2. 3に FORTRANモデルラッピング用の要素モデル開発者が、開発時に意識すべき内容を示します。

まず、CommonMPの要素モデルの親クラスの派生として ラッパー要素モデルを作成します。

次に、上記ラッパー要素モデルに FORTRANサブルーティンへの引数値を設定・取得する処理を実装します。

また、同様に FORTRANサブルーティンをコールする処理を実装します。

FORTRANモデル開発者が行う内容

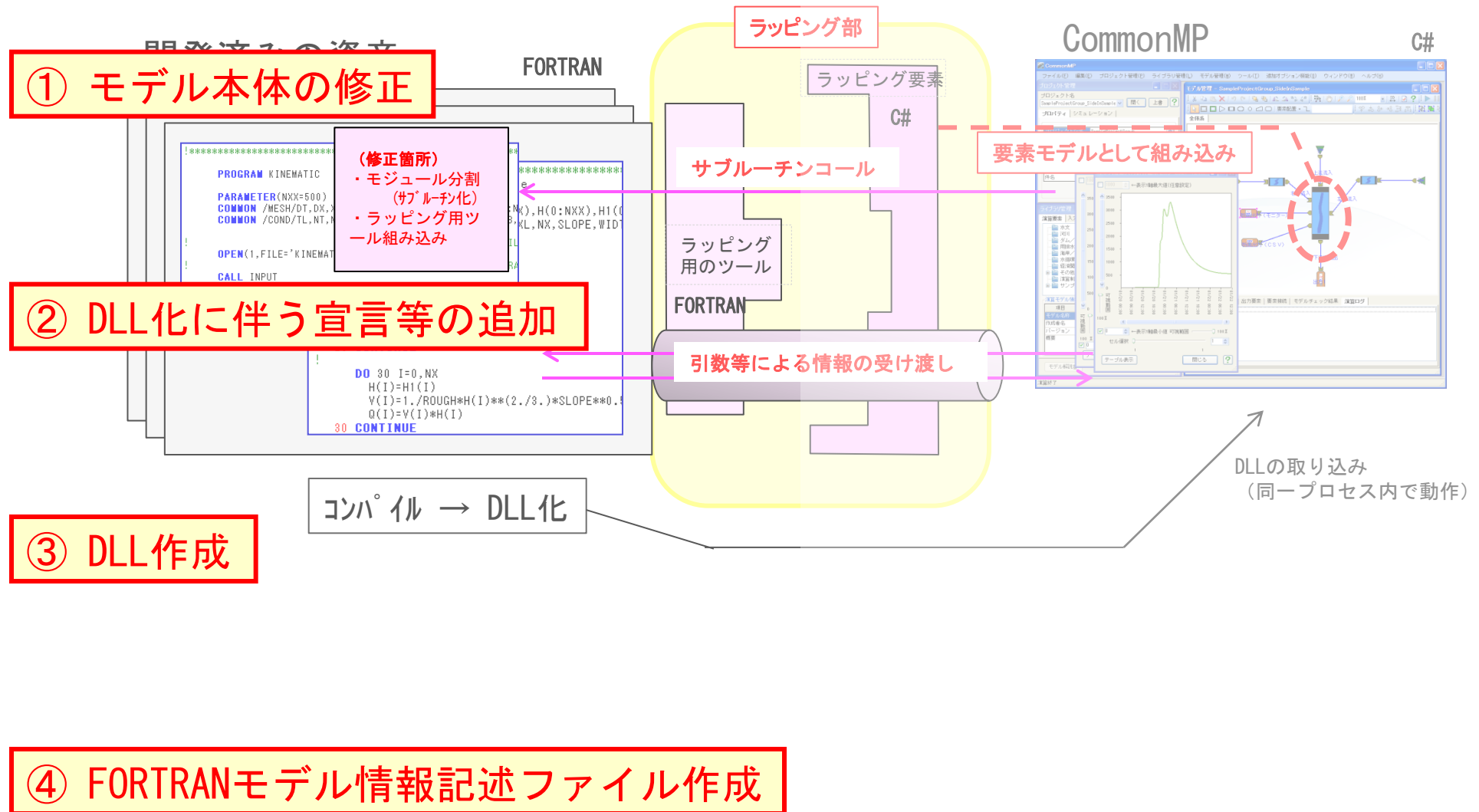


図2. 2 モデル開発者（FORTRAN使用）にとって、ラッピングの為に行う作業

ラッパー要素モデル開発者が行う内容

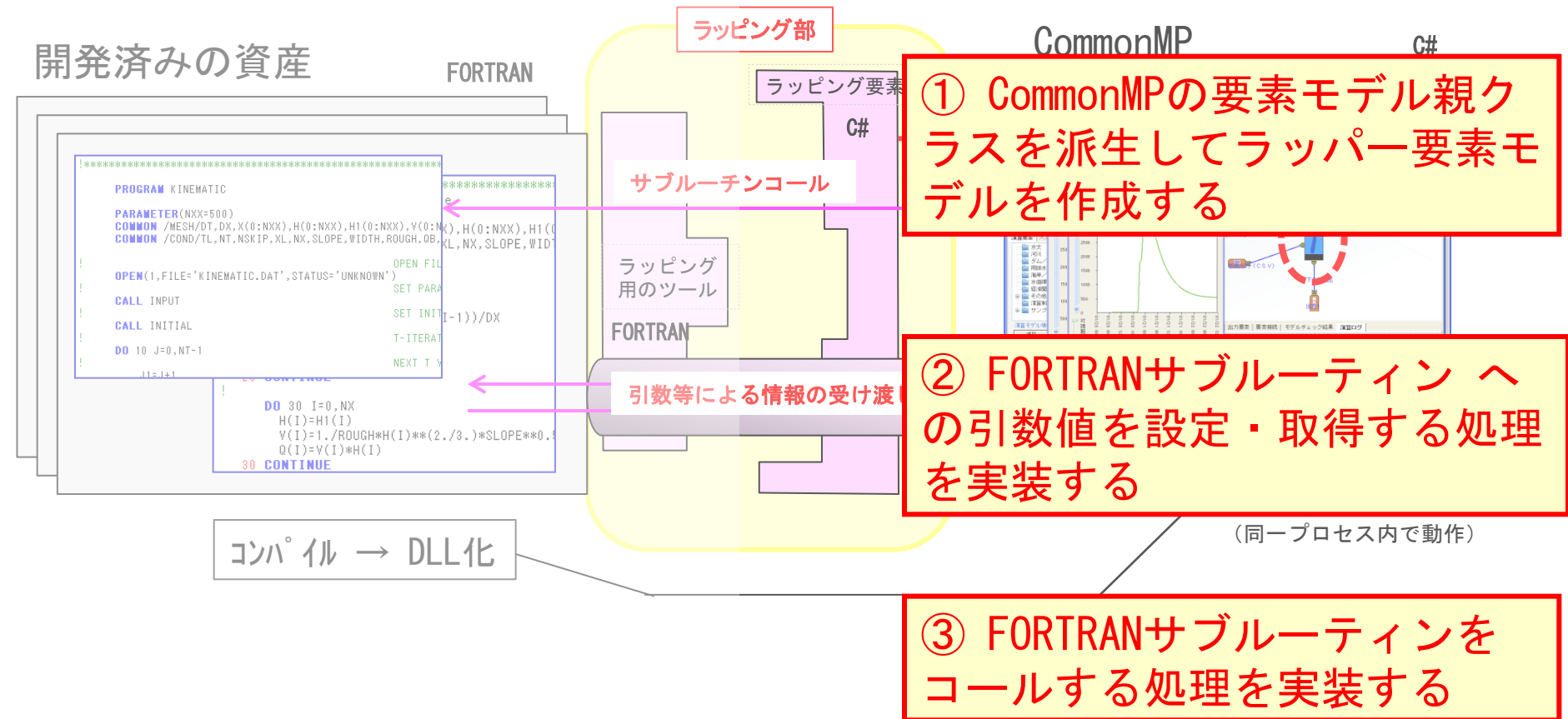


図 2. 3 FORTRANモデルのラッピング用要素モデル開発者が行う作業

3. サンプルの動作概要

本サンプルが FORTRANモデルを 要素モデルとして取り込む処理の概要を 図3. 1に示します。 FORTRANモデルラッピング要素モデルは1種類作成し、モデル情報を記述したファイルを読み込むことで、同一の仕様で作成された 複数のFORTRANモデルに対応する事ができます。

FORTRAN モデル作成者は FORTRANモデルをコンパイルして「DLL」を作成すると同時に、その DLL名や引数等に関する「モデル情報記述ファイル」を作成します。ここで作成した「DLL」および「モデル情報記述ファイル」を

¥CommonMP¥Execute¥bin¥FortranModelDll¥

下に置きます。

本サンプルでは、FORTRANモデルのDLLを開発する環境として インテル社製のVisualFortranComposer (Windows版)と、マイクロソフト社のVisualStudio2005 を使用しています。(いずれも32ビット版)

また、CommonMP対応前のオリジナルFORTRANプログラムとして、土木学会偏 水理公式集 (平成11年版) 例題プログラム集 第2編 河川編 例題：2－2 不定流計算からKinematic Wave, Dynamic Wave のFORTRANプログラムを参考にしました。

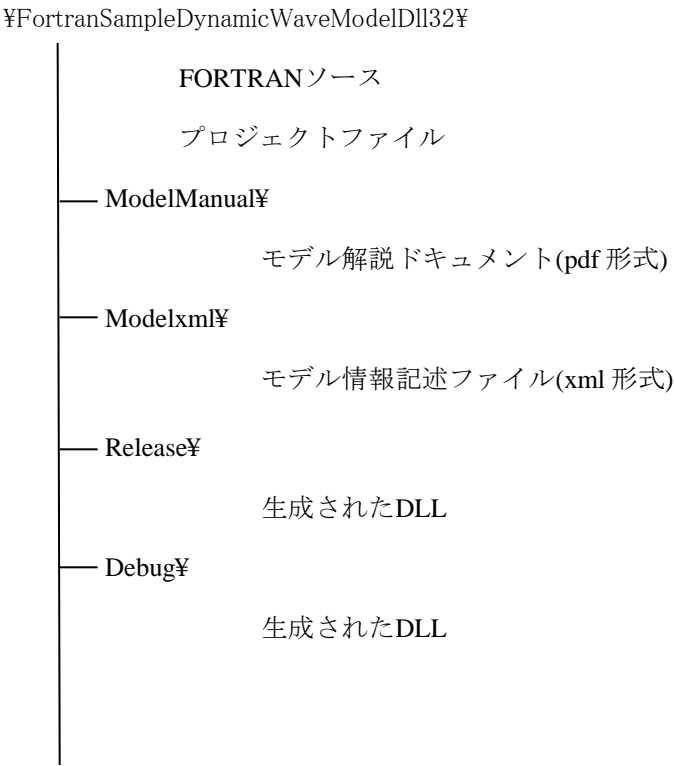
FORTRANモデルの「DLL」作成プロジェクトは

¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDvlp¥

下のディレクトリ「FortranSampleDynamicWaveModelDll32」 および、「FortranSampleKinematicWaveModelDll32」の中に置いてあります。

(図3. 2参照)

各モデルの開発プロジェクトのディレクトリ下には 下に示すような 各種ファイルおよびディレクトリが存在します。



新しくFORTRANプログラムを作成する場合のプロジェクト型紙が 同ディレクトリ下の McFortranModelTPL として置いてあります。この中には CommonMP用のDLL作成に必要なプログラムやスタブが一式含まれており、これをコピーして使用すると、容易にモデル実装用のプロジェクトが作成できると思います。

「モデル情報記述ファイル」はXML形式で記述します。本ファイルの作成には通常のテキストエディターを使用しても可能ですが、XMLフィルの構造を意識しなくても「モデル情報記述ファイル」が作成できるように ¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDvlp¥Tools¥McFortranInfoEditor¥ 下に 簡易なエディターを作成しました。(エディタープログラムはC#で作成しています) この簡易エディターは ソースを各自でコンパイルして使用する事を前提としていますが、 コンパイルしないで使用したい場合には、 ¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDvlp¥Tools¥McFortranInfoEditor¥bin¥Release¥ 下の 「McFortranInfoEditor.exe」と 「Mc32BitFortranDLLWrapper.dll」 を、 ¥CommonMP¥Execute¥bin¥ にコピーし、そこから 「McFortranInfoEditor.exe」を ダブルクリックして エディターを起動してください(図3. 2参照)。本エディターはコーディング例を示すためのサンプルなので、正常処理しか実装されていません。

一方、FORTRANモデルラッピング用の要素モデルサンプル および、同プロパティ設定画面は、 通常の要素モデルの開発と同様の手順で行います。

FORTTRANラッピング要素モデル および、プロパティ設定画面の開発プロジェクトはそれぞれ

¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDLLWrapper32¥

¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDLLWrapper32Property¥

に置かれています。

通常の要素モデル開発と同様に、マイクロソフト社製 VisualStudioから ¥CommonMP¥Source¥HYMCO¥OptionImpl¥ModelDeveloperExpressEdition ¥TestModelDeveloperMainExp.sln を開き 内部の動作をトレースする事ができます。また、ここで作成した ラッピング要素モデルのDLLを ¥CommonMP¥Execute¥bin 下にコピーすることで、CommonMP本体から使用できるようになります。(図3. 4参照)

FORTRAN モデル

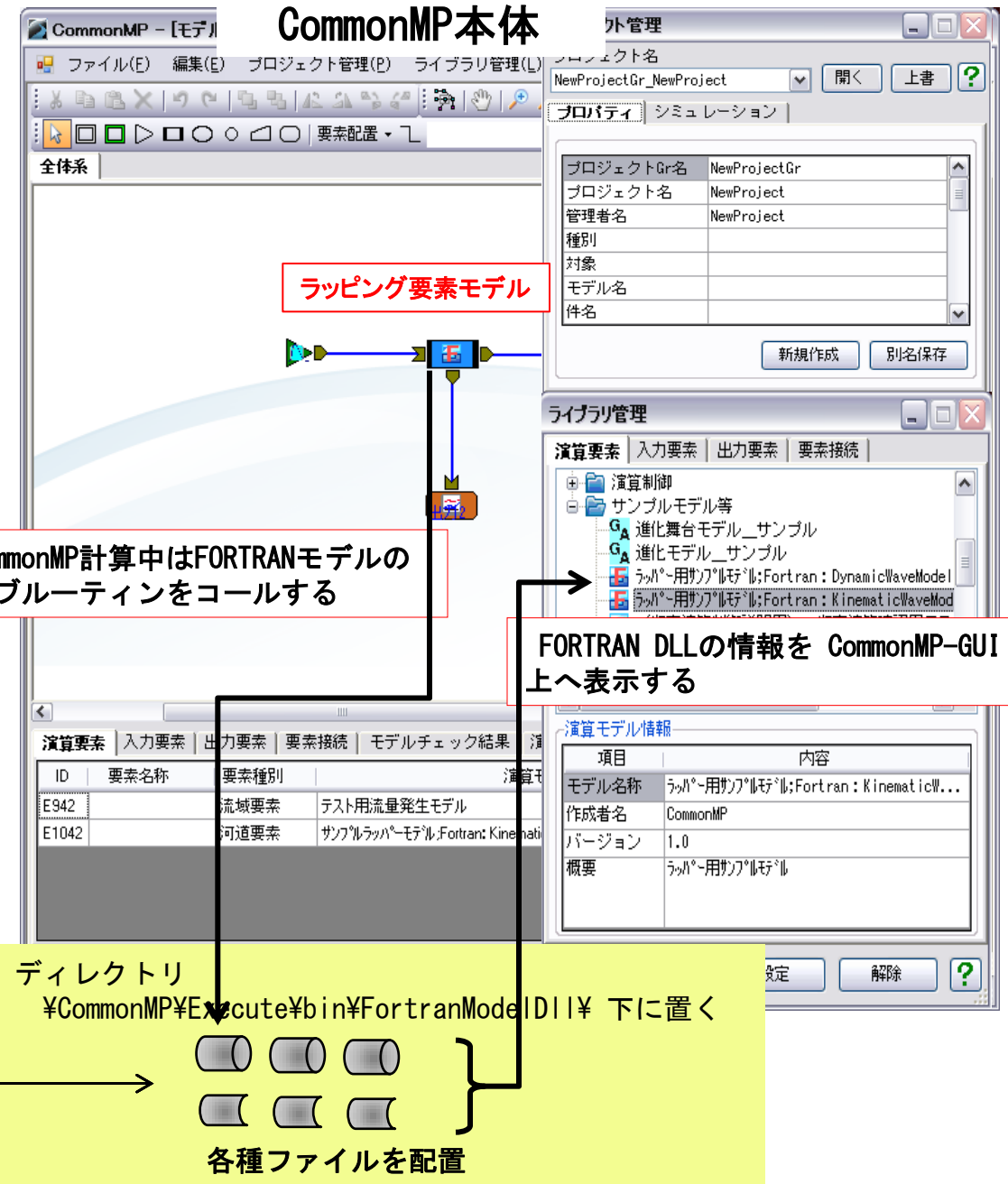
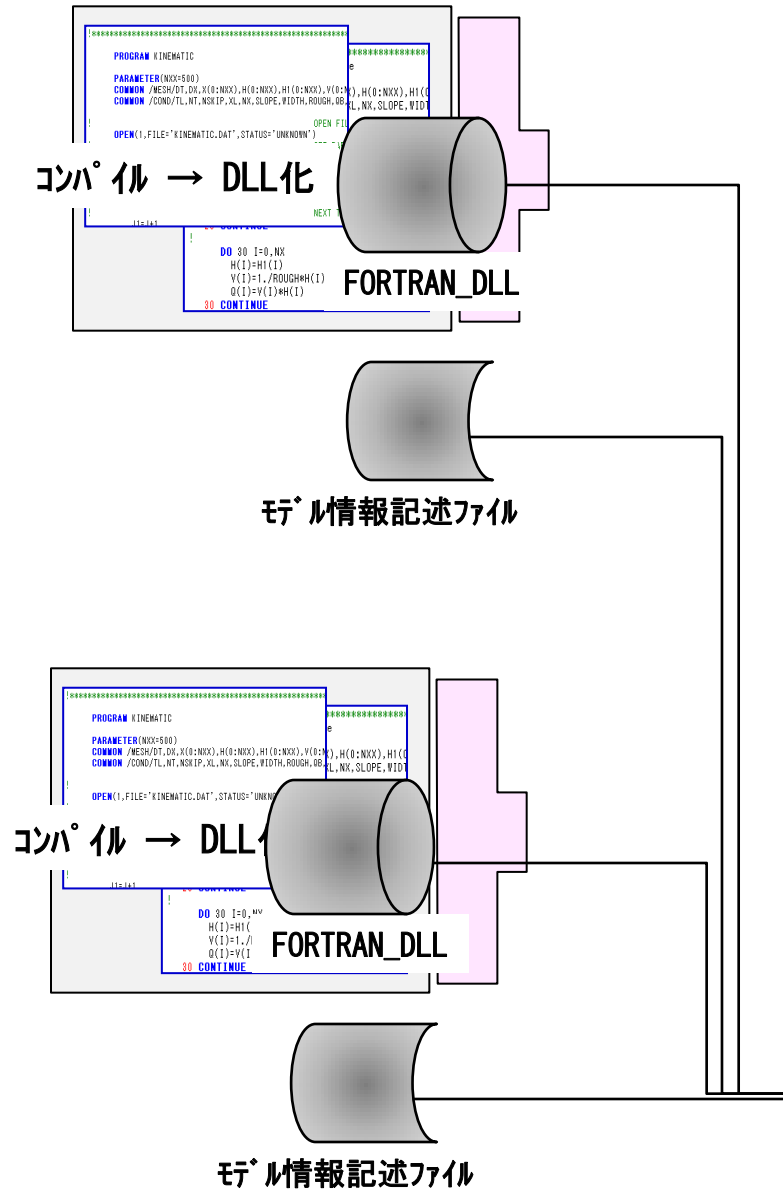
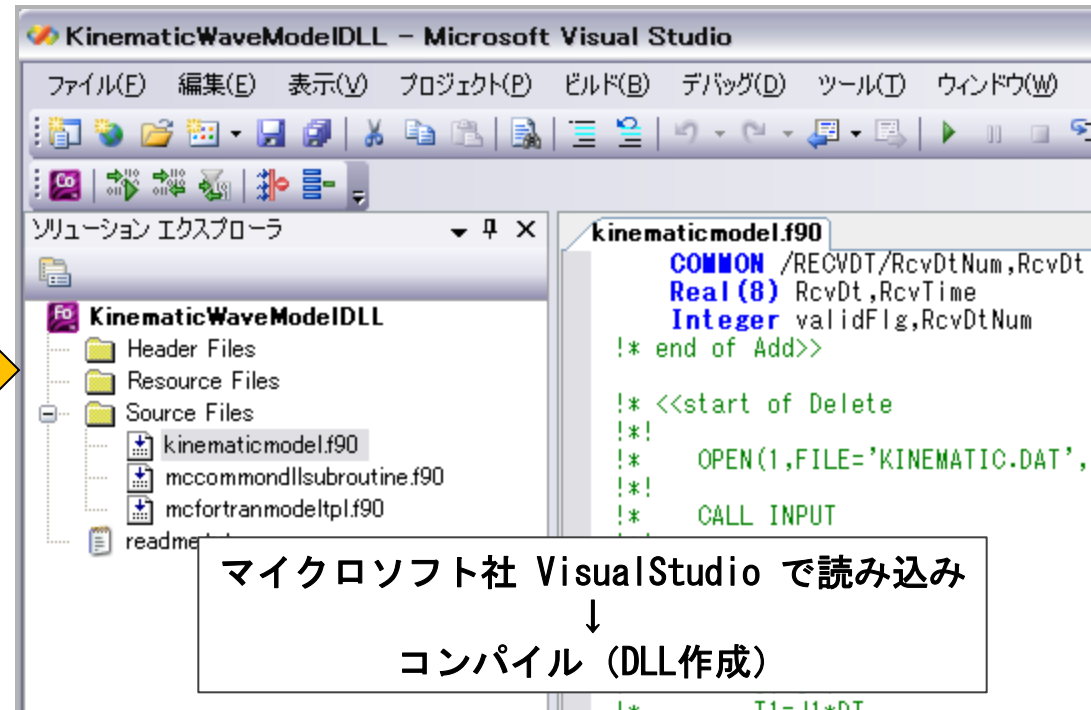
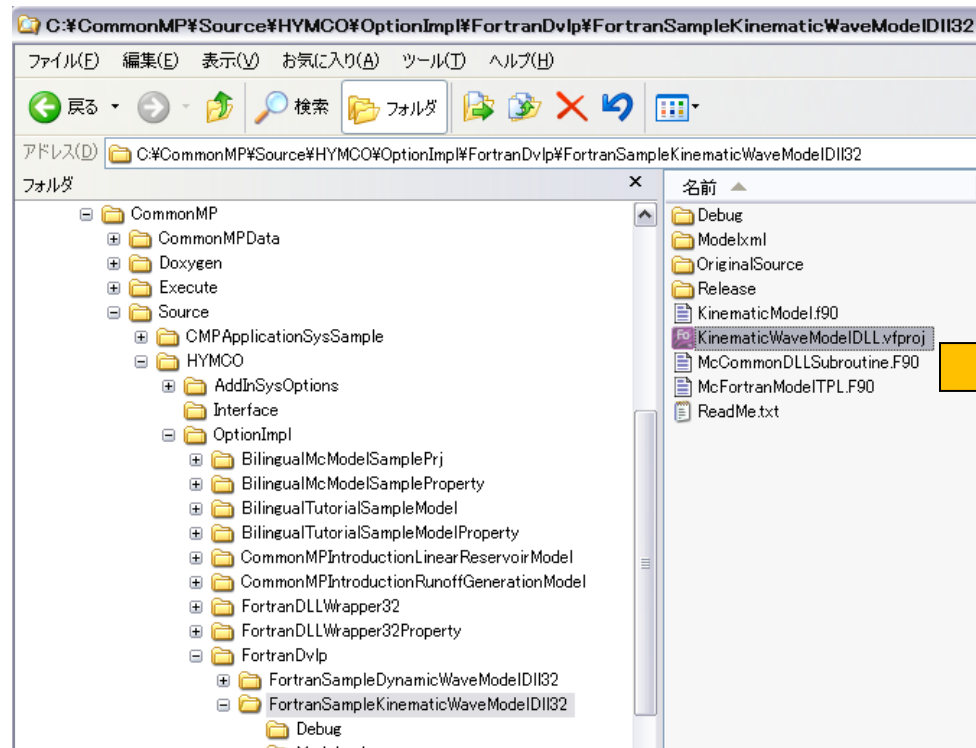


図 3. 1 FORTRANのモデルの組み込み処理概要



FORTTRANサンプルプロジェクト

¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDvlp¥FortranSampleKinematicWaveModelDll32 下

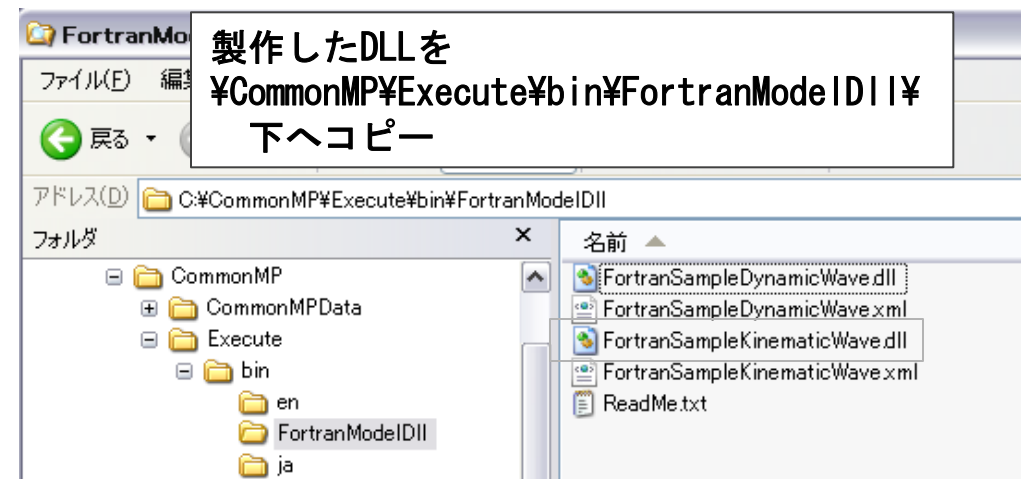


図 3. 2 サンプルFORTTRAN プロジェクト

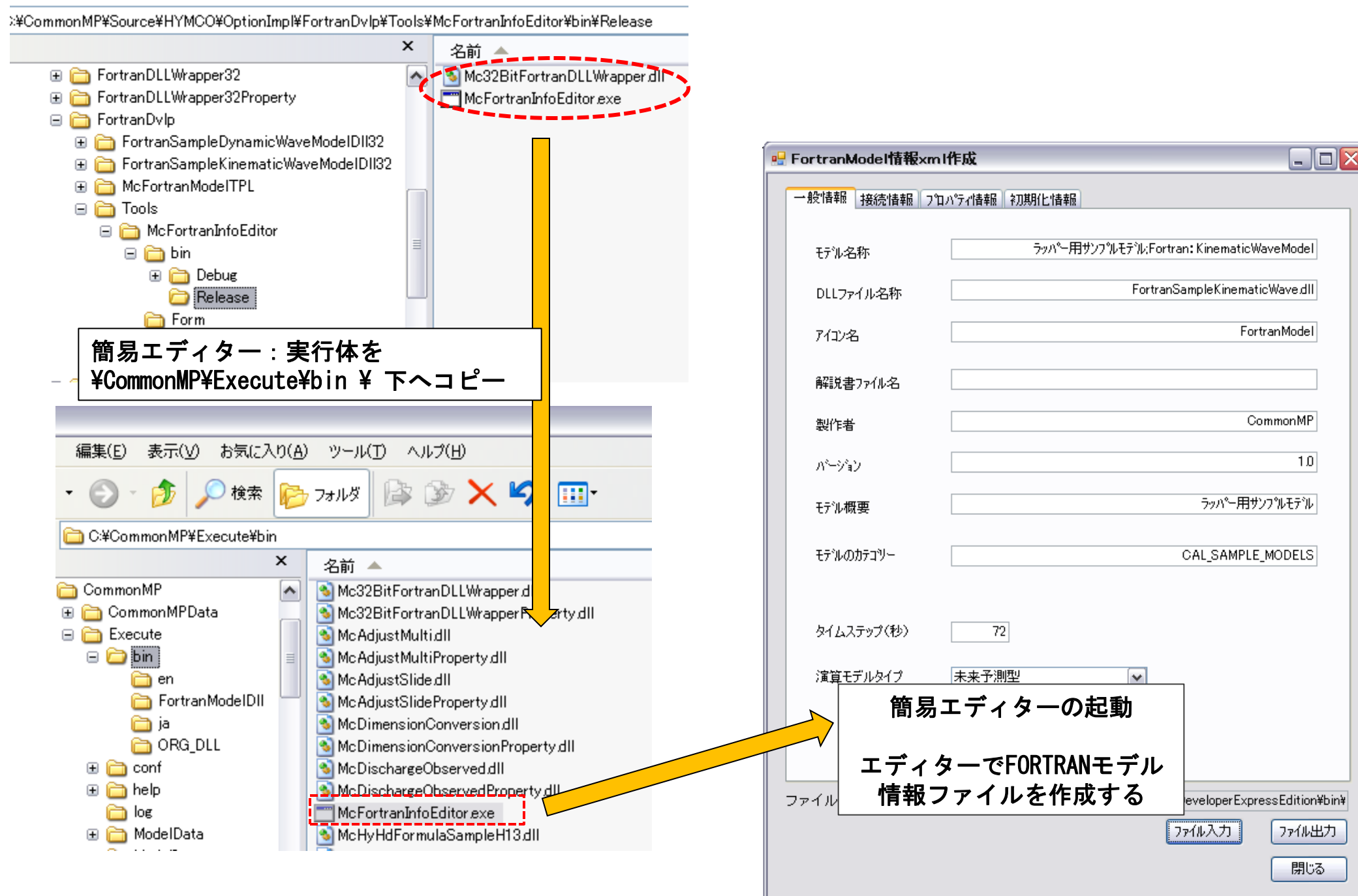
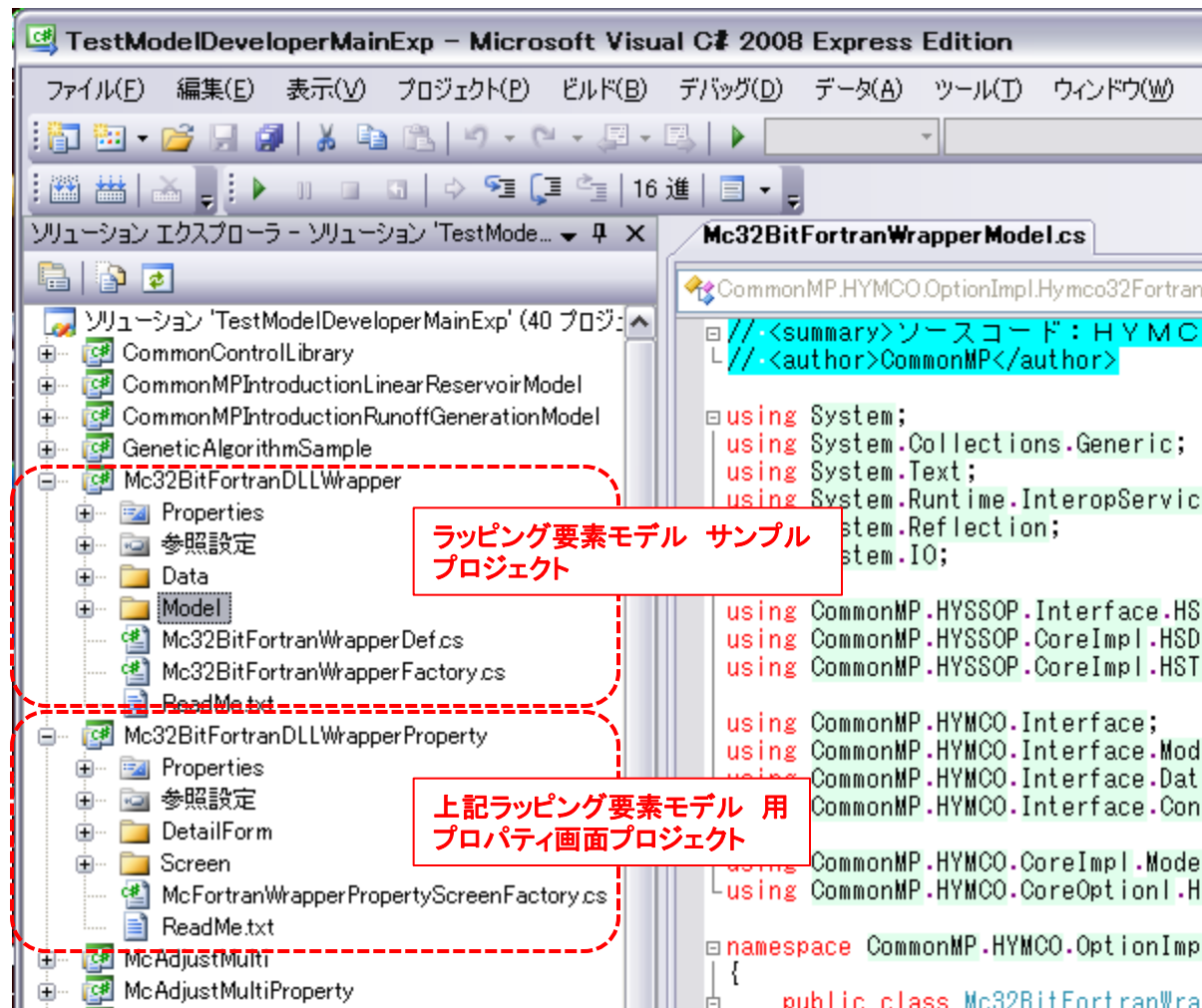


図 3. 3 サンプルFORTRANモデル情報ファイル作成用エディターの起動



製作したラッピング要素モデルのDLLを ¥CommonMP¥Execute¥bin¥ 下に配置する

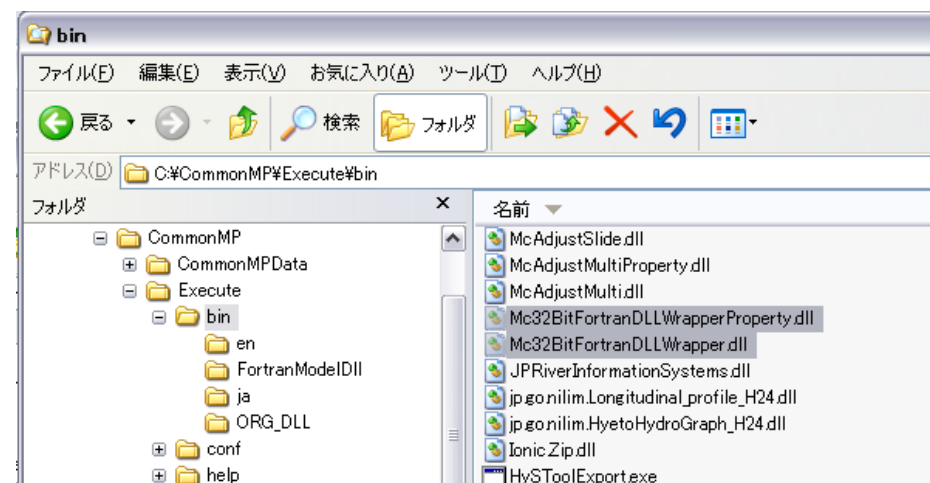
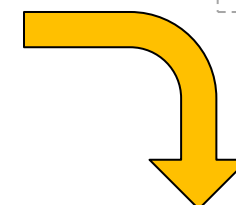


図 3. 4 サンプルFORTRANモデルラッパー用の要素モデル サンプルプログラム

4. FORTRANモデルの作成

本サンプルでは、FORTRANプログラムを サブルーティンとして「DLL」化し、ラッピング要素モデル側から そのサブルーティンをコールすることで あたかも FORTRANモデルを CommonMPの要素モデルの一つとして扱おうというものです。

尚、本サンプルでは、

執筆者：池田裕一先生

土木学会編 水理公式集（平成11年版）

例題プログラム集 第2編 河川編

例題：2-2 不定流計算から

Kinematic Wave, Dynamic Wave のFORTRANプログラム を 参考にして ラッピング用サンプルを作成しています。

4. 1 FORTRANプログラムの修正

2章で簡単に触れたように、本サンプルでは モデルをサブルーティンとして作成する必要があります。（既に開発されたFORTRANプログラムが存在するときには、サブルーティン化する必要があります。）そこで、本サンプルでは FORTRANプログラムの修正の考え方と、必要なサブルーティンについて説明します。

図4. 1の左に示すFORTRAN オリジナルソースは 同図右に示す様に「初期化」、「モデル計算」、「結果出力」等の役割別に 機能ブロックとして分割でき、それらの機能ブロックが一連の流れの中で結合されていることが判ります。ここで分割した処理フローを CommonMPの処理フローと比べて見ると 図4. 2に示すように非常に良く似た構造となっています。そこで FORTRANプログラムを CommonMPで使用するために 図4. 3に示す様にオリジナルソースをCommonMPの処理単位に対応したサブルーティン（またはファンクション）に分割します。

図4. 4に実装が必要なサブルーティン（またはファンクション）を示します。

これらのサブルーティン（またはファンクション）群は 型紙としてスタブが

```
¥CommonMP¥Source¥HYMCO¥OptionImpl¥FortranDvlp¥McFortranModelTPL¥
```

下に McFortranModelTPL.F90 として記述してあります。FORTRANモデルを変更する場合には 本型紙のソースをコピー／ペーストして使用すると手間が軽減されます。

尚、計算終了判断、モデル内の時刻を進める処理は 共通であることが多いため、上記ファイルではなく、 McCommonDLLSubroutine.F90 ファイル内に コーディングしています。

Kinematic Wave, Dynamic Wave の各モデルのオリジナルなソースに対して、上記サブルーティン化等の変更を行ったソースを それぞれ

```
¥ FortranSampleKinematicWaveModelDll32¥KinematicModel.f90
```

```
¥FortranSampleDynamicWaveModelDll32¥dynamic.f90
```

に 置きました。

また、McFortranModelTPL.F90 の型紙内に定義されているものの、モデルの動作にはほとんど関係が無いサブルーティン 例えば、CommonMPのGUI上から計算を一時ストップさせたときにコールされる処理(integer function SuspendCalc())等は、スタブ状態で残しておいて下さい。（CommonMPラッピング要素モデルからのリンクを保つ為）

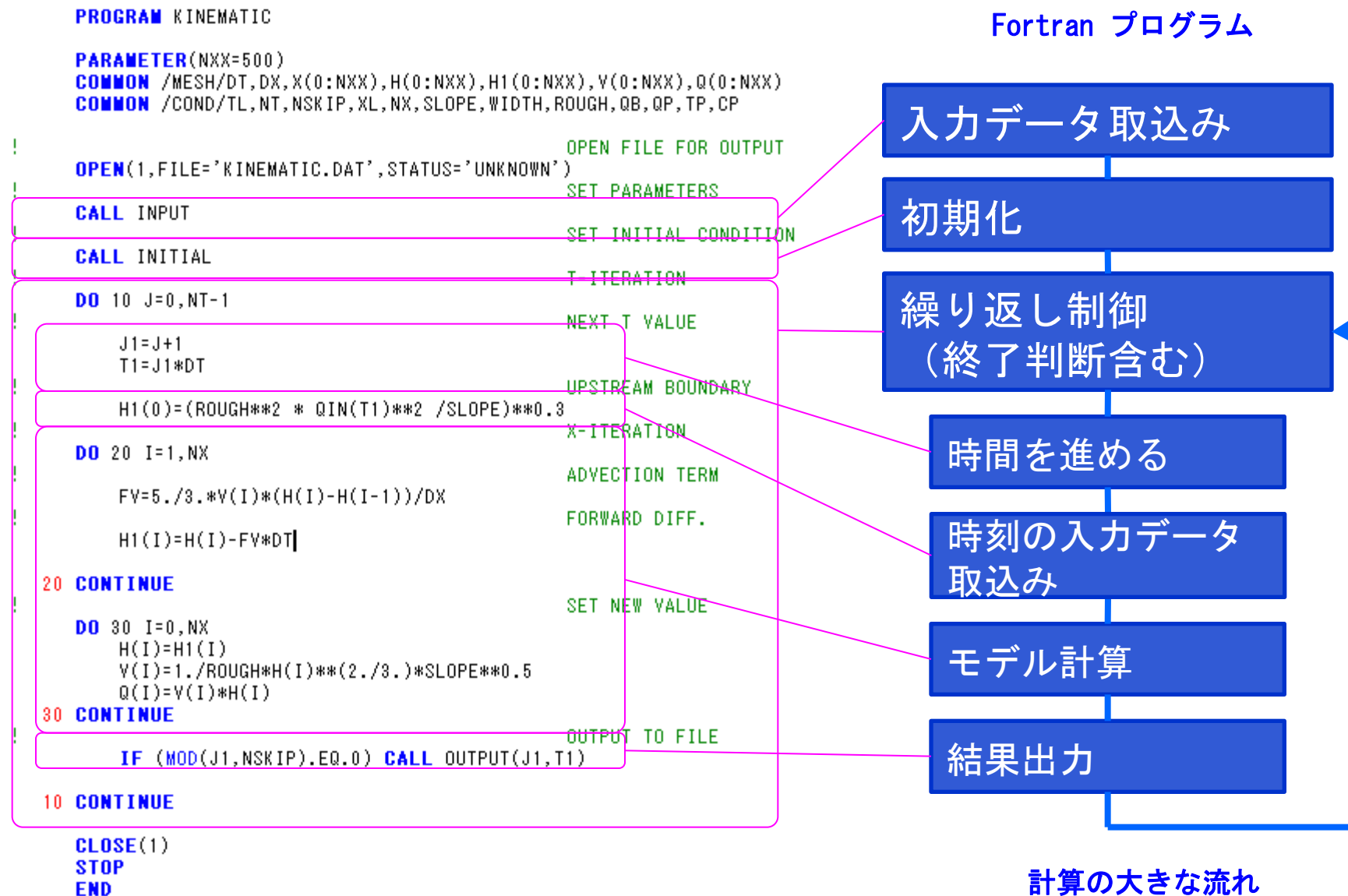


図4. 1 オリジナルなFORTRANプログラムの処理構造

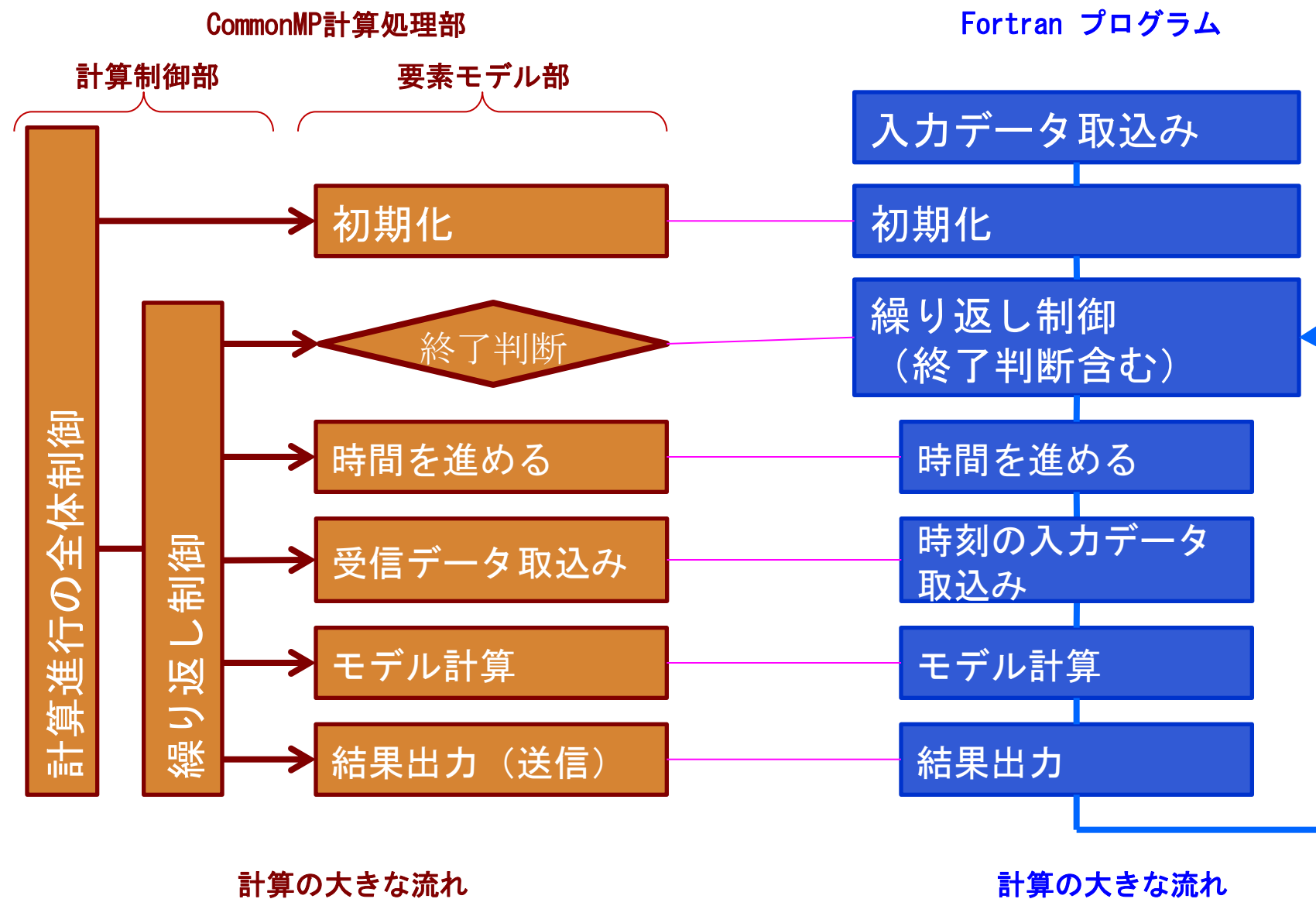
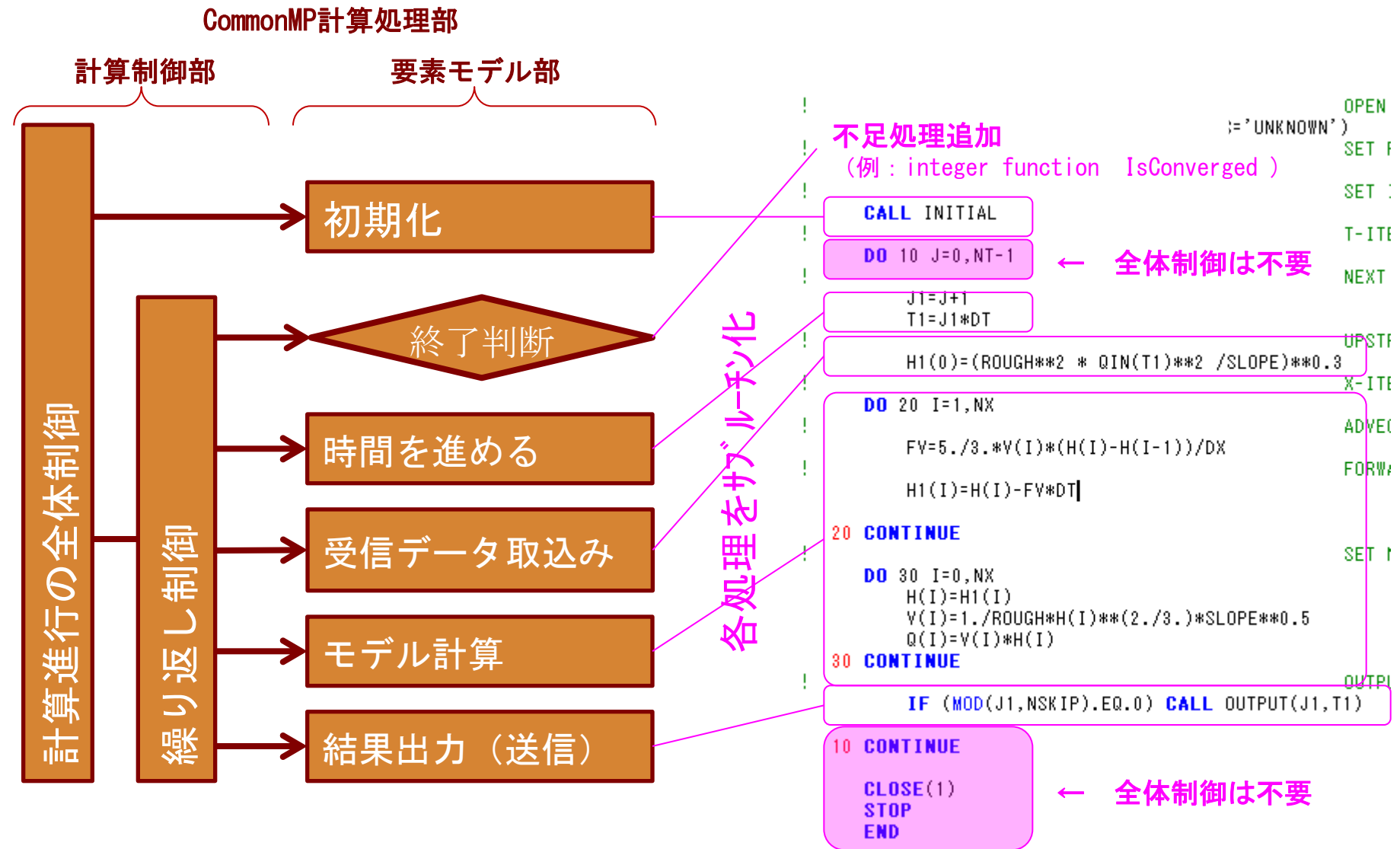


図4. 2 オリジナルプログラムの処理構造とCommonMP処理構造の比較



計算の大きな流れ

図4. 3 オリジナルプログラムのCommonMP化対応のためのサブルーティン化の単位

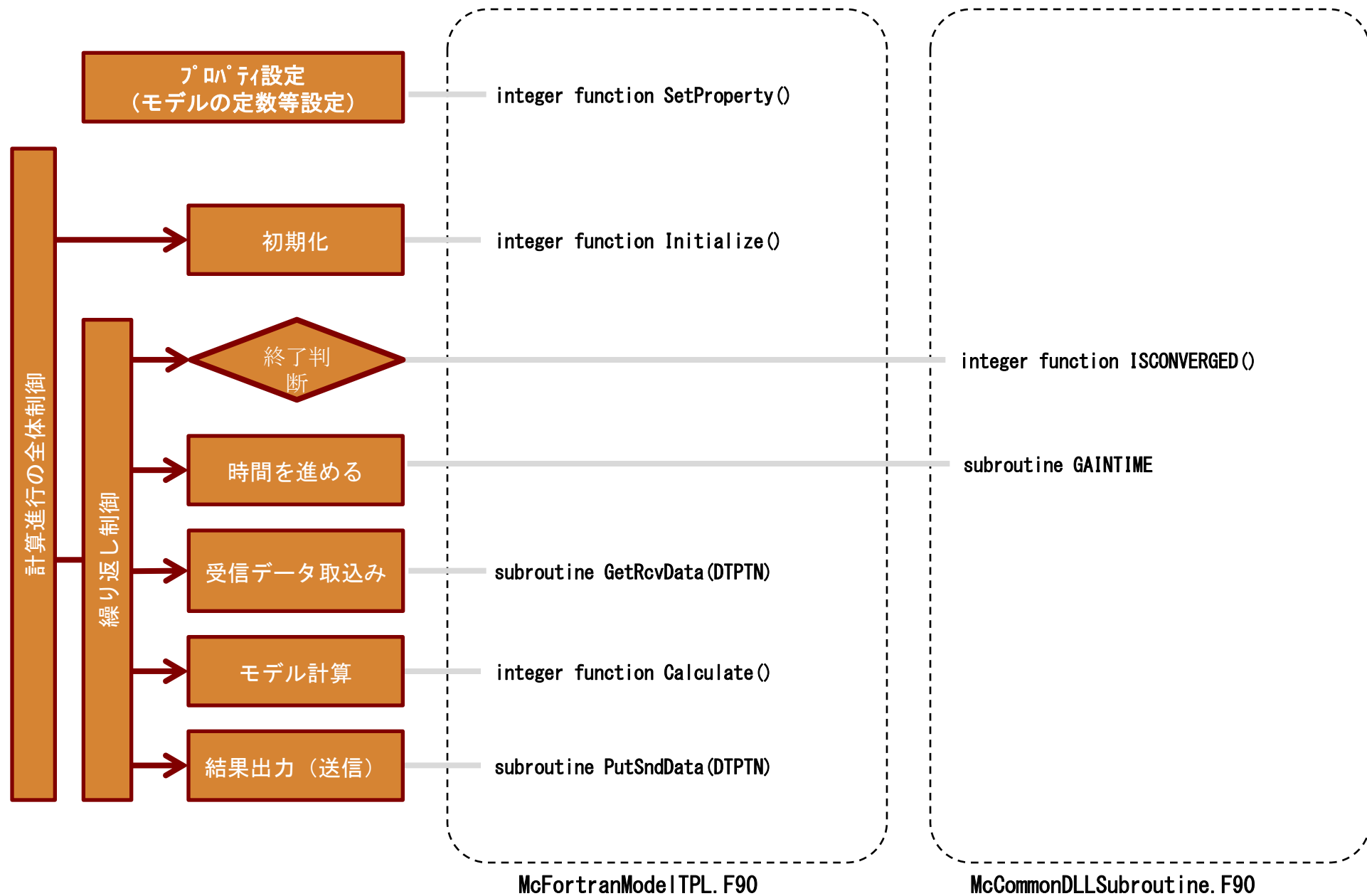


図 4. 4 実装すべきサブルーティン (ファンクション)

4. 2 DLL化に伴う宣言等の追加

C#で実装されたCommonMPから FORTRANで作成されたDLL内のサブルーティンをコールするためには、外部に対して プログラムのエントリーポイントを開放するための宣言(Export)等が必要となります。この宣言はFORTRANコンパイラに依存します。本サンプルではインテル社製のVisualFortranComposer(Windows版)を使用した例を示します。

4. 1章で作成したサブルーティンに対して 直接 DLLの外部宣言(Export)を行うことも可能ですが、先にも述べた通り、これらの宣言はコンパイラに依存しており、コンパイラを変更した場合には それに伴ってソースの修正が発生する可能性があります。そこで、図4. 5に示すとおり、DLL化に伴う宣言を一つにまとめ、ここから、4. 1章で作成したサブルーティンと呼ぶようにしてあります。この様にすることで、FORTRANモデル開発者はモデルの論理だけに集中できるとともに、コンパイラ等の変更があっても修正するファイルを一箇所に集めることができます。本サンプルでは DLL化に関連する処理を McCommonDLLSubroutine.F90 に纏めています。

<サブルーティン間の情報伝達>

本サンプルでは、上記の処理構造において、各サブルーティン間の情報に COMMON文を使用しています。COMMON文の使用については批判もありますが、FORTRANはオブジェクト指向言語とは異なり、情報の隠蔽という概念が文法上保障されていないため、本サンプルでは積極的に利用しています。

使用するCOMMON領域の全変数は McCommonDLLSubroutine.F90
ファイル内サブルーティン

```
subroutine McCommonSubTPL
```

内に記述しています。

定義された変数を以降に述べます。

subroutine McCommonSubTPL 内で定義されている COMMONエリア

! 配列の最大数

```
integer MAXPARA,MAXIN,MAXOUT  
PARAMETER(MAXPARA=64)  
PARAMETER(MAXIN=500)  
PARAMETER(MAXOUT=500)
```

! モデルのプロパティ (パラメータ) 情報格納領域

! モデルの初期値 (パラメータ) 情報格納領域

```
COMMON /PRPTY/ LParaNum, LPara(0:MAXPARA), DParaNum, DPara(0:MAXPARA)  
COMMON /INITVAL/ LValNum, LVal(0:MAXPARA), DValNum, DVal(0:MAXPARA)  
Integer LParaNum, LPara, DParaNum, LValNum, LVal, DValNum  
Real(8) DPara, DVal
```

! モデル内の時間 (シミュレーション時刻、計算目標時刻、 δT)

```
COMMON /CTL/SimTime,TargetTime,deltaT  
Real(8) SimTime,TargetTime,deltaT
```

! 外部からの受信情報 (受信配列 (生) データ)

```
COMMON /RECVDT/RcvDtNum,RcvDt(0:MAXIN),RcvTime,validFlg  
Real(8) RcvDt,RcvTime  
Integer validFlg,RcvDtNum
```

! 計算結果送信情報 (送信配列 (生) データ)

```
COMMON /SENDDT/SndDtNum,SndDt(0:MAXOUT),SndTime  
Real(8) SndDt,SndTime  
Integer SndDtNum
```

CommonMP計算処理部

C#

初期化

終了判断

時間を進める

受信データ取込み

モデル計算

結果出力（配信）

コール

コール

FORTAN

```

計算実行処理
integer function CALMODEL()
!DEC$ ATTRIBUTES DLLEXPORT::CALMODEL
!DEC$ ATTRIBUTES STDCALL::CALMODEL
!DEC$ ATTRIBUTES REFERENCE::CALMODEL
CALMODEL=Calculate()
end function CALMODEL

計算結果送信処理
subroutine SENDDATA(LNUM, DOUT, DOUTTM, DTF
DEC$ ATTRIBUTES DLLEXPORT::SENDDATA
DEC$ ATTRIBUTES STDCALL::SENDDATA
DEC$ ATTRIBUTES ALIAS:'FSetSndData'
DEC$ ATTRIBUTES REFERENCE::LNUM
DEC$ ATTRIBUTES REFERENCE::DOUT
DEC$ ATTRIBUTES REFERENCE::DOUTTM
DEC$ ATTRIBUTES REFERENCE::DTF
implicit none
integer MAXOUT
PARAMETER (MAXOUT=500)
! データパターン別にCommon領域にデータを記
CALL PutSndData(DTPTN(0))
! Common領域の情報を 引数に設定する
LNUM(0) = SndDtNum;
DOUTTM(0) = SndTime;
DO 10 J=0,SndDtNum-1
DOUT(J) = SndDt(J)
10 CONTINUE
end subroutine SENDDATA
    
```

McCommonDLLSubroutine.F90

コール

コール

FORTAN

```

integer function Calculate()
!*PROGRAM KINEMATIC
integer MAXIN
PARAMETER (MAXIN=500)
!* end of Modify>>

PARAMETER (NXX=500)
COMMON /MESH/DT,DX,X(0:NXX),H(0:NXX),H1,
COMMON /COND/TL,NT,NSKIP,XL,NX,SLOPE,WIC

subroutine PutSndData(DTPTN)
Integer DTPTN
!* end of Modify>>

!* <<start of Add
integer MAXOUT
PARAMETER (MAXOUT=500)
COMMON /CTL/SimTime,TargetTime,de
Real(8) SimTime,TargetTime,deltaT
COMMON /SENDDT/SndDtNum,SndDt(0:MAXOUT)
Real(8) SndDt,SndTime
Integer SndDtNum
!* end of Add>>

PARAMETER (NXX=500)
COMMON /MESH/DT,DX,X(0:NXX),H(0:NXX),H1,
    
```

McFortranModelTPL.F90
KinematicModel.f90

図4. 4 DLL化に伴う宣言等を記述した共通サブルーティンと モデル開発者が作成するサブルーティンの関係

```
integer MAXPARA, MAXIN, MAXOUT
      PARAMETER (MAXPARA=64)
      PARAMETER (MAXIN=500)
      PARAMETER (MAXOUT=500)
```

は 配列数を定義しています。この配列数は、それぞれ
McCommonDLLSubroutine.F90 ファイル内に定義されている

```
integer function MAXPARAMUM ()
```

```
integer function MAXRCVDIM ()
```

```
integer function MAXSNDDIM ()
```

により CommonMPのラッピング要素モデルに取り込まれて、内部でのメモリ確保に使用されます。

```
COMMON /PRPTY/ LParaNum, LPara(0:MAXPARA), DParaNum, DPara(0:MAXPARA)
COMMON /INITVAL/ LValNum, LVal(0:MAXPARA), DValNum, DVal(0:MAXPARA)
Integer LParaNum, LPara, DParaNum, LValNum, LVal, DValNum
Real(8) DPara, Dval
```

はCommonMPのラッピング要素モデルの画面から入力したプロパティ情報、初期化情報をFORTRANプログラムがCommonMP側から受け取る時に使用する領域です。これらの情報はモデルによって異なりますが、ここでは Integer 型配列、Real(8) 型配列を 定義して使用します。配列の何処にどのような変数が入るかは 各モデルによって自由に使用します。

・ COMMON /PRPTY/ プロパティ情報

LParaNum : Integer型変数の数

LPara : Integer型変数 (配列)

DParaNum: Real (8) 型変数の数

DPara : Real (8) 型変数 (配列)

・ COMMON /INITVAL/ 初期化情報

LValNum : Integer型変数の数

LVal : Integer型変数 (配列)

DValNum: Real (8) 型変数の数

DVal : Real (8) 型変数 (配列)

```
COMMON /CTL/SimTime, TargetTime, deltaT
Real(8) SimTime, TargetTime, deltaT
```

は モデル内の時間を管理する変数でそれぞれ シミュレーション時刻、計算を行う目標時刻、 δT を意味します。

```
COMMON /RECVDT/RcvDtNum, RcvDt(0:MAXIN), RcvTime, validFlg
Real(8) RcvDt, RcvTime
Integer validFlg, RcvDtNum
```

は モデルに接続された他の要素モデルから 本モデルに対して送られてきた情報を表し、Real(8)型変数配列で渡されます。

```
COMMON /SENDDT/SndDtNum, SndDt(0:MAXOUT), SndTime
Real(8) SndDt, SndTime
Integer SndDtNum
```

は 本モデルから他のモデルへ送信する情報の格納領域を表し Real(8)型変数配列として定義されます。

CommonMPでは 他の要素モデルとの送受信に使用する情報は 図4. 5に示すように 流量、水位、流速等を一つにまとめて「セル」として扱い、その「セル」を1次元、2次元、3次元配列として伝送します。本サンプルでは、これらのセル配列を、図4. 6に示す様に 1次元のReal(8)型変数の配列に変換して渡します。FORTRANモデル開発者は 各セルの情報がどのように 1次元のReal(8)型変数の配列に配置されるかを知っておく必要があります。各セルに どのような情報が配置されるかは、「モデル情報記述ファイル」に定義しておくことで、FORTRANモデルとの整合を取ることが出来ます(4. 4章参照)。

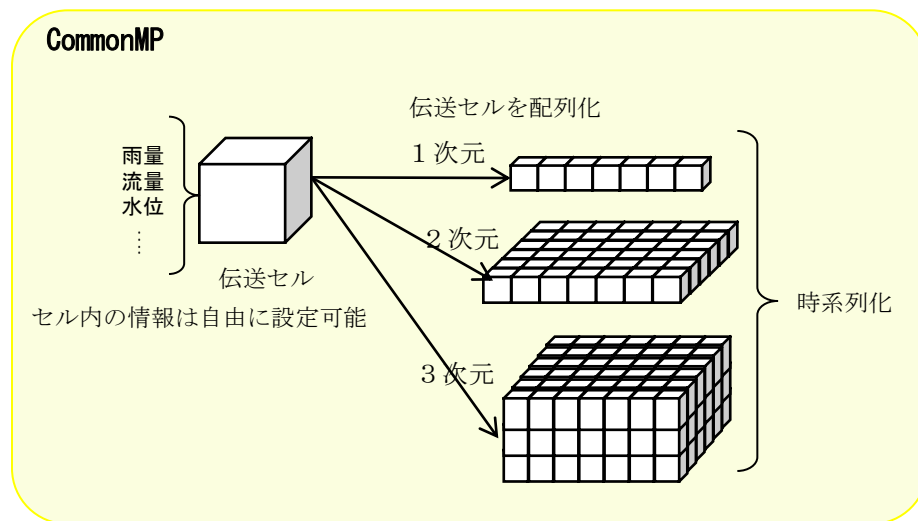


図 4. 5 CommonMPのモデル間伝送仕様（概念）

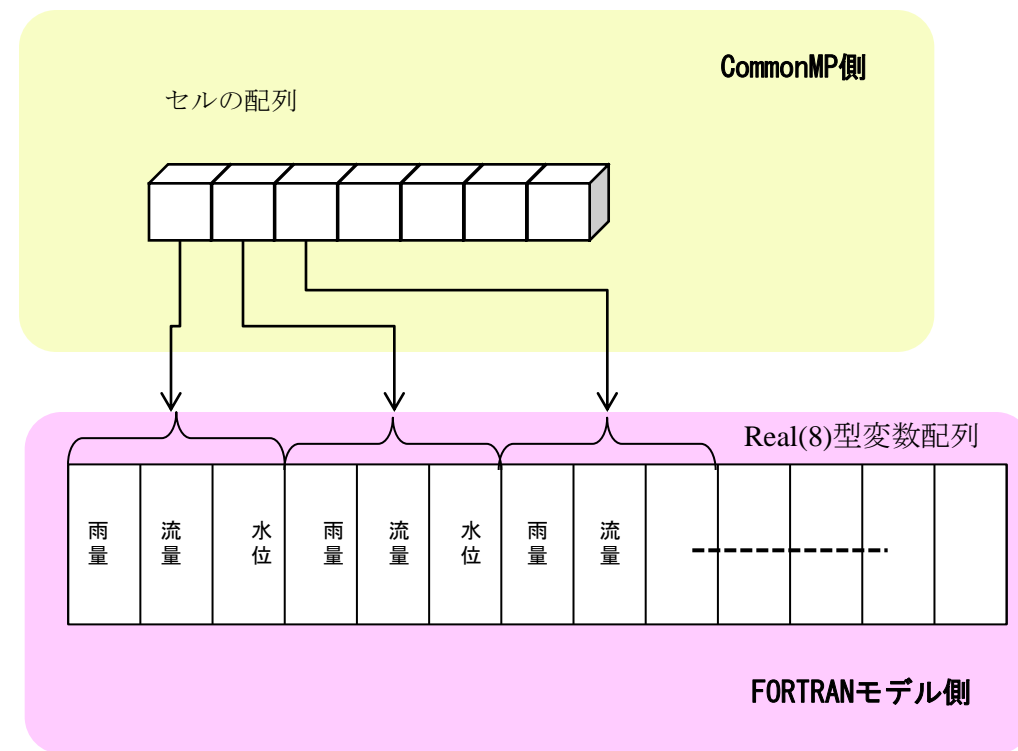


図 4. 6 セル配列と一次変数配列の関係

4. 3 DLLの作成

4. 1章で作成したプログラムと、4. 2章で説明した DLL用の共通サブルーティンを纏めて コンパイラーリンクし DLLを作成します。

左記のコンパイルによって生成されたDLLを
¥CommonMP¥Execute¥bin¥FortranModelDll¥ 下に コピーします。

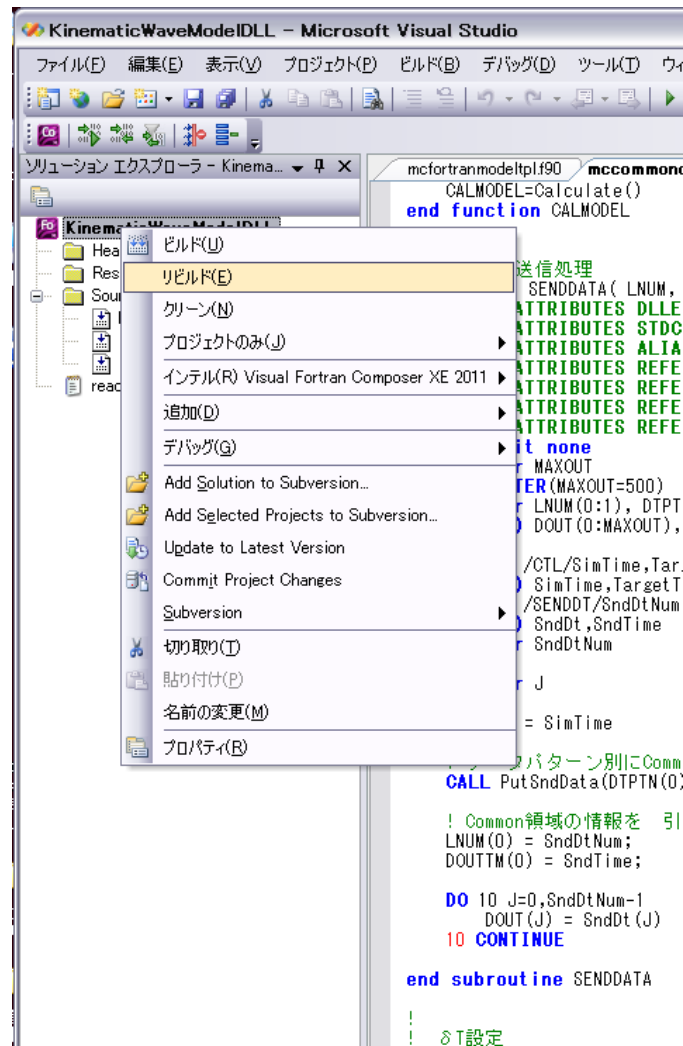


図 4. 7 コンパイルの一例

4. 4 FORTRANモデル情報記述ファイル作成

CommonMPでは 要素モデルのプロパティ情報(定数などの設定情報)や初期化情報等を 画面から設定し、プログラムを修正することなくモデルの特性を変更することが可能です。しかしながら、FORTRANモデル側で直接画面を呼び出すことはできないので、本サンプルでは、CommonMPのGUI上に表示する内容や、オペレーターが設定した内容とFORTRANプログラムの対応を定義した「モデル情報記述ファイル」を通じて、オペレーターのGUI上での設定値を FORTRANモデルに引数値として与えます。この「モデル情報記述ファイル」はXML形式で記述します。本サンプルでは「モデル情報記述ファイル」を編集する簡易なエディターを 提供します。エディター自身もサンプルであるため、最低限の処理のみ記述してあります。独自のプログラムを作成するときの参考としてください。

図4. 8(a)～図4. 8(d)に エディター画面の設定と FORTRANプログラム内の情報、および CommonMP-GUI上に表示される情報の関係を示します。

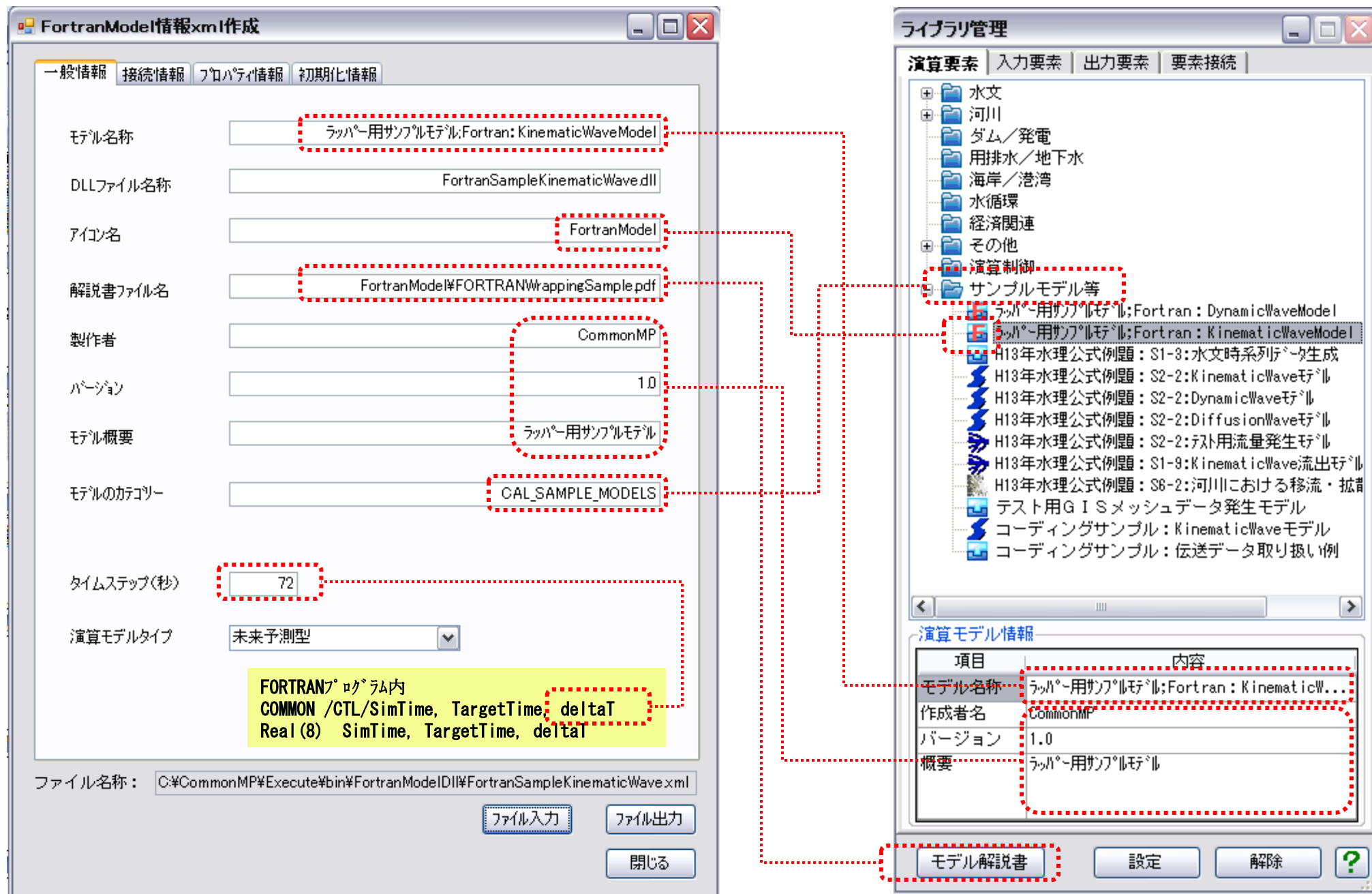


図 4. 8 (a) モデル情報記述ファイルエディターとCommonMP上の画面表示の対応（一般情報）

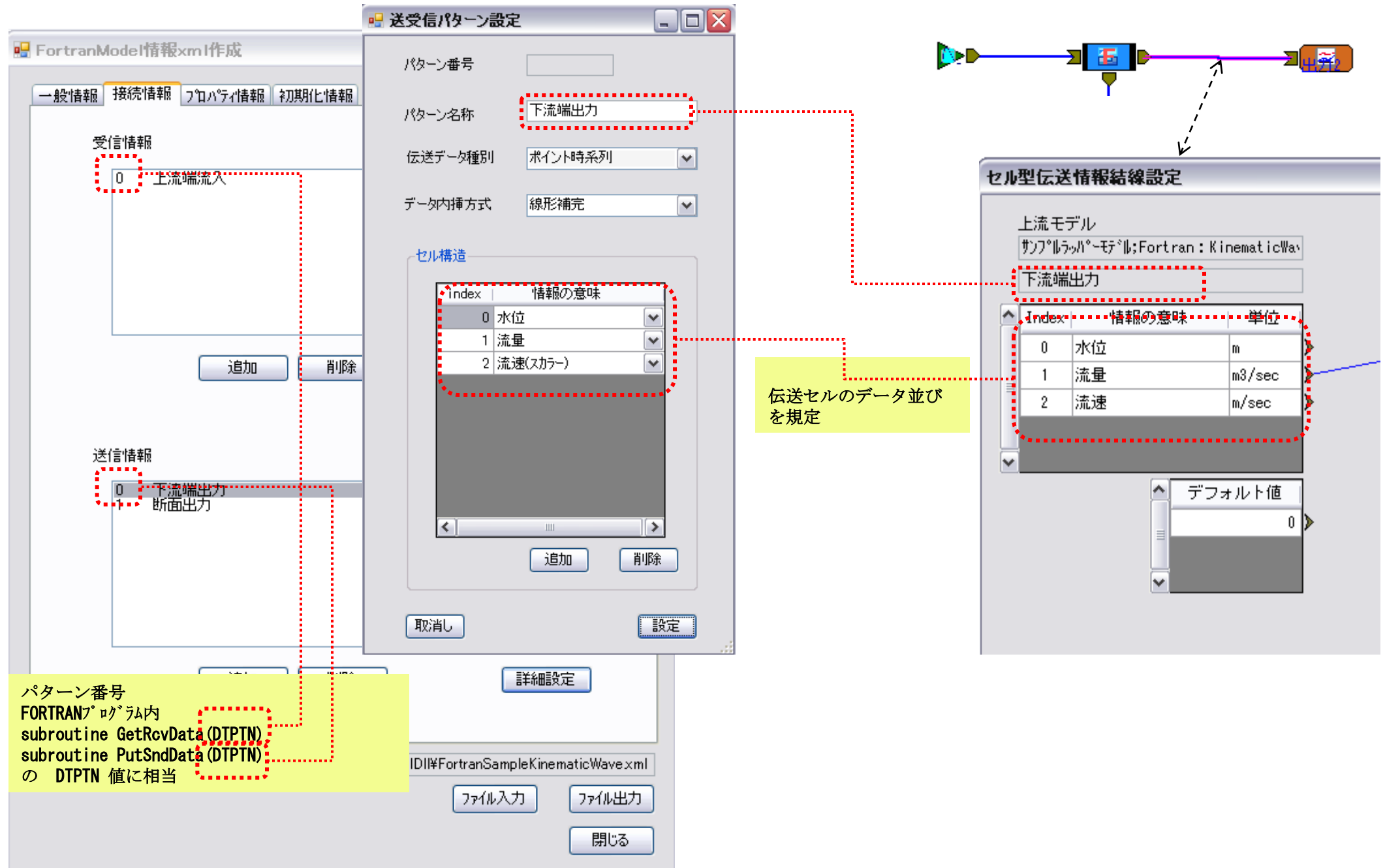


図 4. 8 (b) モデル情報記述ファイルエディターとCommonMP上の画面表示の対応（接続情報）

FortranModel情報xml作成

一般情報 接続情報 プロパティ情報 初期化情報

Integer型変数 1 個

追加 削除

データ型	項目番号	項目	値
Integer	L0	モデル内セル分割数+1	51

Fortranプログラム内
COMMON /PRPTY/ LParaNum に相当

Real(8)型変数 4 個

追加 削除

データ型	項目番号	項目	値
Integer	L0	粗度	0.03
Integer	L1	川幅[m]	200
Integer	L2	川長[m]	100000
Integer	L3	河床勾配	0.0005

Fortranプログラム内
COMMON /PRPTY/ DParaNum に相当

ファイル名称: C:\CommonMP\Execute\bin\FortranModelID1\FortranSampleKinematicWave.xml

ファイル入力 ファイル出力 開じる

CommonMP伝送プロパティ画面

フォートランラッパーモデルプロパティ詳細設定画面

デフォルトタイプ 未来予測型

データ型	項目番号	項目	値
Integer	L0	モデル内セル分割数+1	51
Real(8)	D0	粗度	0.03
Real(8)	D1	川幅[m]	200
Real(8)	D2	川長[m]	100000
Real(8)	D3	河床勾配	0.0005

Fortranプログラム内
COMMON /PRPTY/ LPara(0:MAXPARA) に相当
(項目番号は配列の位置を示す)

Fortranプログラム内
COMMON /PRPTY/ DPara(0:MAXPARA) に相当
(項目番号は配列の位置を示す)

ファイル出力 設定 キャンセル ?

図 4. 8 (c) モデル情報記述ファイルエディターとCommonMP上の画面表示の対応 (プロパティ情報)

FortranModel情報xml作成

一般情報 接続情報 プロパティ情報 初期化情報

Integer型変数 0 個

追加 削除

データ型 項目番号 項目 値

Fortranプログラム内
COMMON /INITVAL/ LValNum に相当

Real(8)型変数 3 個

追加 削除

データ型 項目番号 項目 値

Integer	L0	初期水位	1
Integer	L1	初期流量	0.1
Integer	L2	初期流速	0.1

Fortranプログラム内
COMMON /INITVAL/ DValNum に相当

ファイル名称: C:\CommonMP\Execute\bin\FortranModel\DI\FortranSampleKinematicWave.xml

ファイル入力 ファイル出力 開じる

CommonMP伝送初期情報画面

プロパティ設定 初期情報設定

設定値

データ型	項目番号	項目	値
Real(8)	D0	初期水位	1
Real(8)	D1	初期流量	0.1
Real(8)	D2	初期流速	0.1

Fortranプログラム内
COMMON /INITVAL/ LVal(0:MAXPARA) に相当
(項目番号は配列の位置を示す)
本サンプルでは、Integer型変数の個数はゼロ (項目が無し)

Fortranプログラム内
COMMON /INITVAL/ DVal(0:MAXPARA) に相当
(項目番号は配列の位置を示す)

ファイル名称:

ファイル入力 ファイル出力 設定 キャンセル ?

図 4. 8 (d) モデル情報記述ファイルエディターとCommonMP上の画面表示の対応 (初期情報)

4. 4 FORTRAN用ラッピング要素モデル

ラッピング用要素モデルは、CommonMPからは、要素モデルとして扱われます。そのため、ラッピング要素モデルは、McCalModel の派生クラスとして作成します。一般の要素モデルは未来予測型として動作する場合は、McForecastModelBase から、現状状態計算型として動作する場合には、McStateCalModelBase から派生させますが、FORTRANモデルがどの型で動作するか不明の為、本サンプルでは 前記要素モデルの親クラスである「McBasicCalculateModelBase」クラスから派生した「Mc32BitWrapperModel」クラスを親として作成します。

要素モデル開発者が実装すべきメソッドは通常のモデルとほぼ同じですが、FORTRAN_DLL で外部にエクスポートされた関数を、内部にインポートする処理や、接続された要素モデルからの送受信情報をFORTRANのサブルーティンの引数に変換する処理等を実装する必要があります。本サンプルでは これらラッピング特有の処理を「DLL演算モデル制御クラス」(Mc32BitFortranModelDLLImport) に纏めました。FORTRAN等で作成されたDLLと C#で生成されたCommonMP間で情報を授受する場合には、アンマネージ領域を使用します。また、CLR (Common Language Runtime) を使用しないC++ DLLでも同様のアンマネージ領域を使用します。そこで、アンマネージ領域を扱う可能性のある処理を「DLL演算モデル制御クラス」に纏めることで、使用する言語やOSの対応ビットによる修正が発生する可能性のある処理を内部に隠蔽しています。

図4. 9にFORTRANモデルラッピング要素モデルのサンプルに関連するクラスの体系を示します。

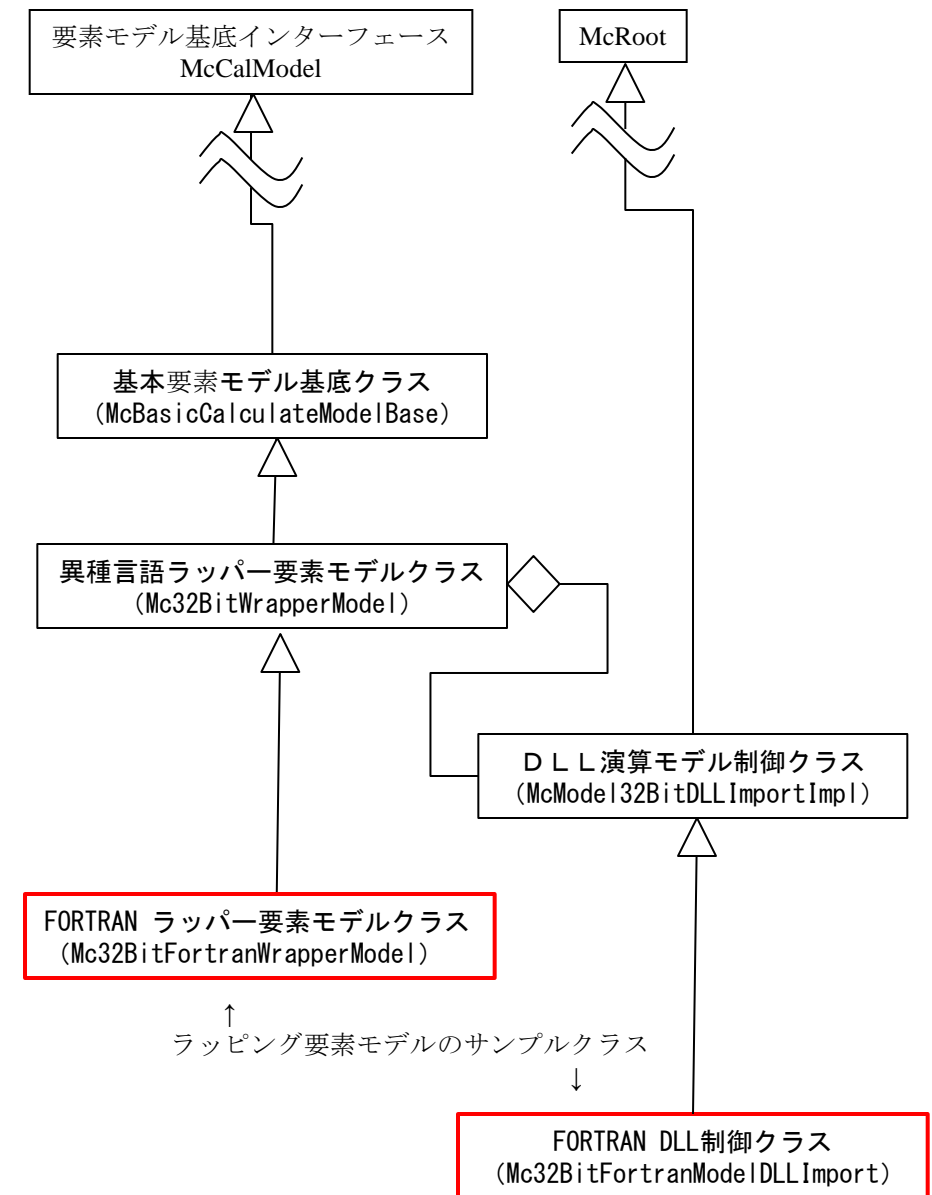


図4. 9 ラッピング要素関連のサンプルクラスの体系図

図4. 10に 前述の「アンマネージ領域」の概念説明図を示します。アンマネージ領域は、C#側からも FORTRAN側からも直接変数として参照できない領域です。従って、C#-FORTRAN間で情報を遣り取りする場合には、変数の値を この領域への情報コピーする操作を行う必要があります。また、この領域を確保するのも 要素モデル作成者(プログラマー)が行う必要があります。これらのアンマネージ領域を扱う処理を「DLL演算モデル制御クラス」内で行っています。

(一方、FORTRAN側では、「McCommonDLLSubroutine.F90」内に関連する宣言等を記述していますので、DLL作成時には このファイルも一緒にコンパイルします。)

アンマネージ領域を通じての変数遣り取りでは、メモリーコピー処理が入るため 大容量の情報を授受する場合には その部分での処理時間が大きくなる可能性があります。

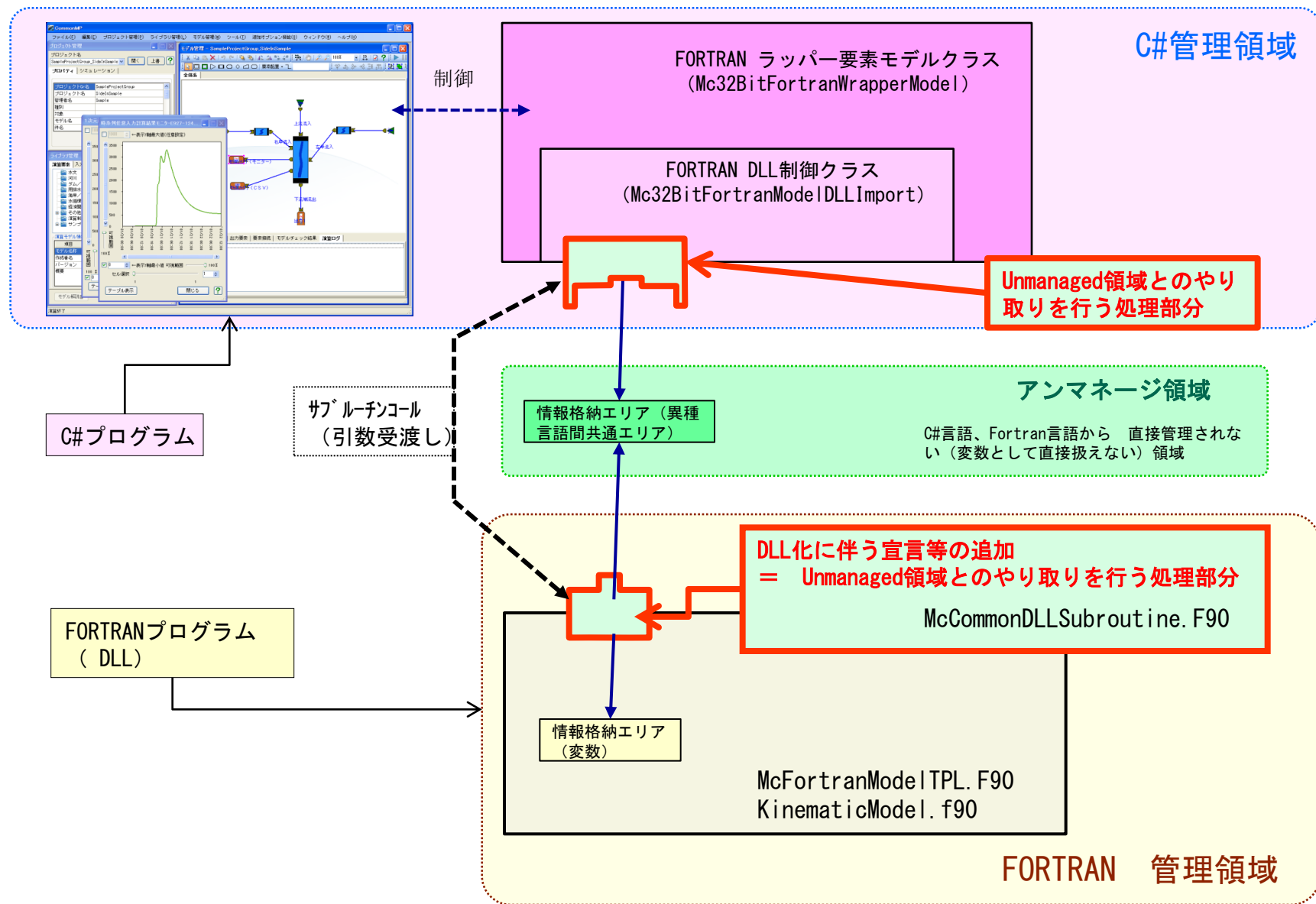


図 4. 10 C#とFORTRANの間での情報受け渡し (概念図)

5. 本ラッピングサンプルにおける限界

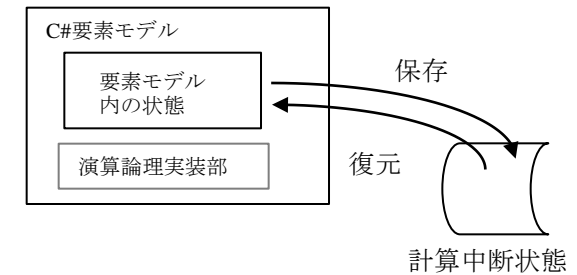
本サンプルのFORTRANモデルラッピング用要素モデルでは 次を示すような制約があります。

1) 1つのDLLで提供可能な要素モデルは1つのみ

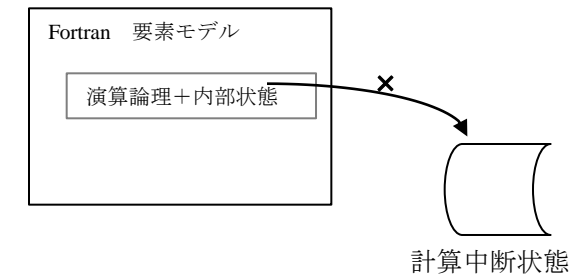
C#でDLLを要素モデルを作成する場合には、ファクトリークラスを作成して、一つのDLLで関連する複数の要素モデルを提供する事が可能でしたが、本サンプルでは、一つのDLL内で一つのモデルしか提供できません。

2) 計算中の中断状態保存は不可能

C#で作成した要素モデルは、計算中の情報を一つのクラスとして纏めている為、(提供された要素モデルが基本構造に従った実装がなされている場合には) 計算中に、一時中断した結果をファイルとして保存しておき、別の機会にそのファイルを読み込んで計算を再開することが可能です。しかしながら、FORTRANモデルでは、計算中の状態を一つのクラスに纏めるという設計思想が無いため、中断状態をファイルに保存しておき、別な機会にそのファイルを読み込んで計算を再開する事は不可能です。(図5. 1参照)



C# 要素モデル



状態（変数）だけを保存する事が事実上不能

FORTRAN モデル

図5. 1 計算中の状態保存

3) 一つのプロジェクト内では 同一のFORTRANモデルを 複数個使用できない

C#で作成された要素モデルは プロジェクト内に複数個 同時に配置し、計算させることが可能ですが、同一のFORTRANモデルを 複数個同時に配置して動作させる事が出来ません。(図5. 2)

オブジェクト指向で作成された要素モデルを使用した場合には 同一の要素モデルを プロジェクト内で複数個配置した時、配置された要素モデル毎に「インスタンス」が生成され、それぞれ別のメモリ領域が割り当てられます。

従って、各インスタンスはお互いに影響を及ぼすことなく動作可能です。

しかしながら、オブジェクト指向言語ではないFORTRANで作成されたモデルでは、「インスタンス」という概念が無いため、複数の要素モデルが同一のメモリー領域を参照し お互いに計算結果を破壊する現象が生じます。このため、一つのプロジェクト内に 同一の要素モデルを複数配置することが出来ません(図5. 3参照)。 この現象は、プロジェクトが複数動作した場合でも同様であり、CommonMP Ver1.2で追加した機能である「複数プロジェクトの同時動作」を行う場合にも、 同一要素モデルが 複数のプロジェクトに含まれていない必要があります。

4) FORTRAN DLL内でのプログラム異常の検出は不能の場合がある

FORTRAN内では、配列の境界領域を越えたエリアにアクセスしようとした場合等の異常が発生した場合に「例外」イベントを スローする機能が無いため、CommonMP側で その「例外」をキャッチする事ができません。 従って、FORTRANプログラム内で異常が発生した場合、CommonMPプログラム全体が異常終了する可能性があります。

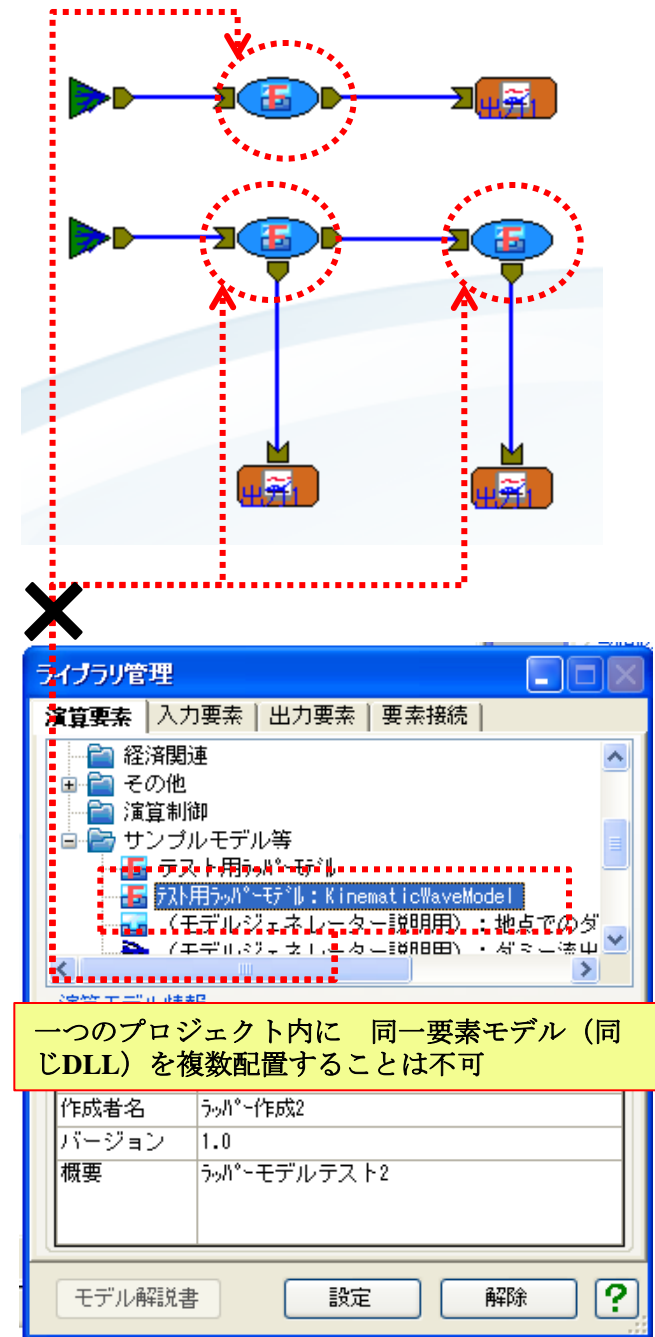
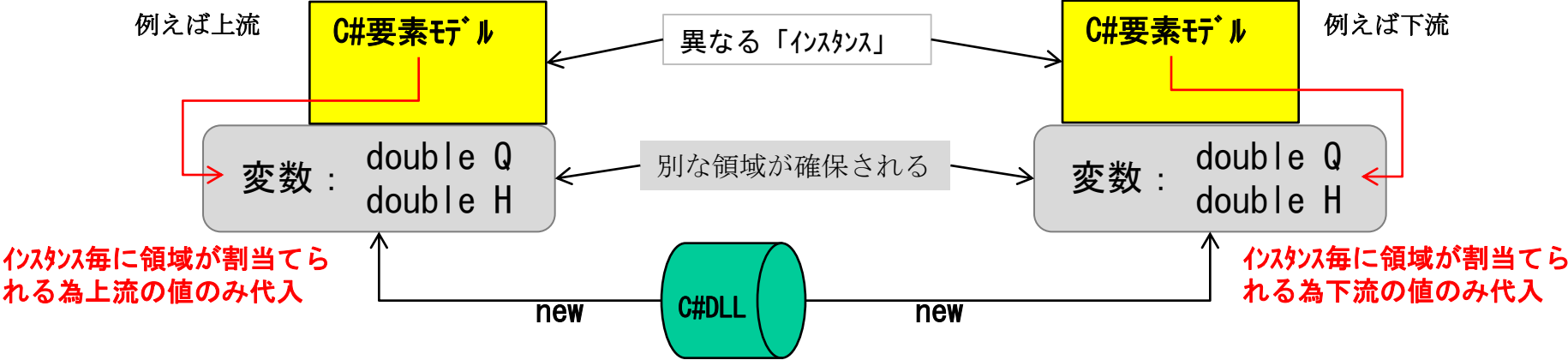


図 5. 2 同一要素モデルの複数配置

●C#(オブジェクト指向言語)の場合



●FORTRANの場合

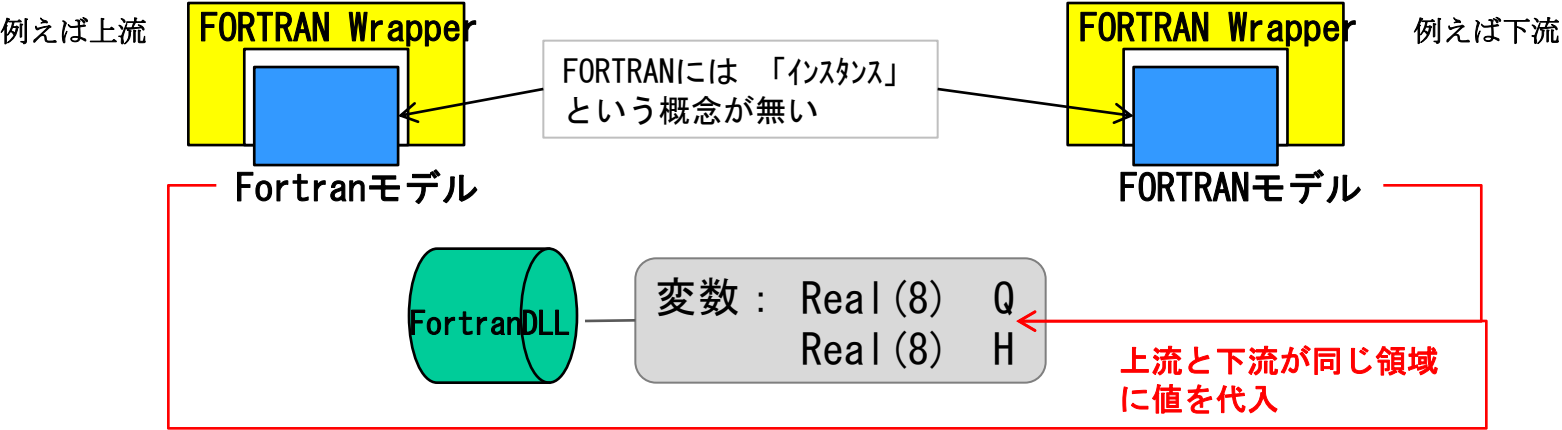


図 5. 3 同一要素モデルの複数配置における動作 (C#とFORTRANの違い)

6. 終わりに

ラッピングについては、既存のFORTRANモデルの処理内容や規模、また、FORTRANモデルをラッピングする必要性や効果を考慮して ラッピングの方法を考える必要があります。

本サンプルの目的は、水理・水門の理論とそれをプログラム化する方法を FORTRANを利用して学習している開発者に対して、C#を意識しなくても比較的CommonMPを利用して頂ける ラッピング要素を サンプルとして提示したものです。従って 本サンプルで紹介した方法では、5章で示した制約事項等が存在しています。また、アンマネージ領域へのメモリコピーなどのオーバーヘッド等も、無視し得なくなる可能性もあります(特に大容量の情報を遣り取りする場合)。

この為、本サンプルは、実用的なラッピング方法を検討する上での参考の一つとして下さい。

<本サンプルの動作範囲>

- 32BitのWindowsOS上で動作する CommonMPに対応します(64Bit環境には対応していません)。

- FORTRANモデルのDLLを開発する環境として インテル社製の VisualFortranComposer (Windows版)と、マイクロソフト社の VisualStudio2005 を使用しています(いずれも32Bit版)。 他の開発環境では動作を試していません。