http://www.gfd-dennou.org

RDoc を用いた数値モデルのドキュメント生成

森川 靖大 (北大理)
石渡 正樹 (北大地球環境)
堀之内 武 (京大生存圏研)
小高 正嗣 (北大理)
林 祥介 (北大理)

はじめに

- ドキュメントの重要性
  - 開発や保守の効率化 (プログラムの改変)
  - ソフトウェアの品質向上 (プログラムの利用)

- Fortran による数値モデルのドキュメント
  - 数理、離散化ドキュメント :: TeX
    - 数式の記述に最適
  - リファレンスマニュアル :: HTML
    - Web からの参照に最適、ハイパーリンクが便利

リファレンスマニュアルの作成

- **何が厄介かというと...**
  - **プログラムとマニュアルを別々に用意するのは面倒**
  - **プログラムで書いたものをもう一度書くのは面倒**
  - Java, Ruby**には**JavaDoc, RDoc**があるが**、Fortran **にはドキュメントを自動生成する標準的な方式が無い**
- **これまでの工夫の一例**
  - XML **を用いた** FMS (GFDL)**の試み**
- **本研究では**
  - RDoc **を用いた** Fortran **ソースコードの自動解析**

XML を用いた FMS (GFDL)の試み

- **Fortran95 に XML のドキュメントを埋め込む**
  - FMS (Flexible Modeling System: GFDL) **で試行**
  - FMS **謹製のツールで** HTML **へ変換**

Fortran95 **ソースコード**

```
module module_name_mod
! <OVERVIEW>
!   module_name_mod の概要
! </OVERVIEW>
  implicit none
  private
  public :: module_name_init, module_name_end
  ! <SUBROUTINE NAME="module_name_init">
  ! <OVERVIEW>
  !   モジュールの初期化
  ! </OVERVIEW>
  ! <TEMPLATE>
  !   module_name_init (inchar, outint)
  ! </TEMPLATE>
  ! <IN NAME="inchar" TYPE="character" >
  !   文字型の入力変数
  ! </IN>
  ! <OUT NAME="outint" TYPE="integer" >
  !   整数型の出力変数
  ! </IN>
  subroutine module_name_init(inchar, outint)
    character(*)    , intent(in) :: inchar
    integer(INTKIND), intent(out) :: outint
  end subroutine module_name_init
  ! </SUBROUTINE>
end module module_name_mod
```

```
! <SUBROUTINE NAME="module_name_init">
! <OVERVIEW>
!   モジュールの初期化
! </OVERVIEW>
! <TEMPLATE>
!   module_name_init (inchar, outint)
! </TEMPLATE>
          :
! </SUBROUTINE>
```

- XML **でコメントを記述する**
- **このコメント部分を抜き出し、**XML **から** HTML **マニュアルを作成**

XML を用いた FMS (GFDL)の試み

- **Fortran95 に XML のドキュメントを埋め込む**
  - **利点**
    - **プログラムとマニュアルを同じファイルで管理**
    - XML **により、構造的なマニュアルを作成可能**
  - **欠点**
    - XML **のタグで** Fortran **コードが汚くなる**
    - XML **の手書きは面倒**
    - **手続き名や引数をコードとマニュアルで** 2 **度書く必要あり**

ソースコード自動解析の必要性

RDoc とは

- 何ぞや？
  - Ruby で書かれたソースコードからドキュメントを自動生成する Ruby の標準ライブラリ
- Fortran 90/95 の解析も可能
  - ソースコード解析機構とマニュアル生成機構が分離しているため、他の言語で書かれたソースコードも解析可能
  - 標準で C および Fortran95 用の解析機構が付属
- 利点
  - FMS と同様な利点を維持し (マニュアルのソースへの埋め込み)、欠点を克服 (タグの簡素化、2度書き解消)

Fortran95 Parser (オリジナル版)

- **Fortran90/95 の文法を解釈**
  - RDoc のタグは XML に比べとても簡潔
  - 別ファイル内のモジュール等へ自動的にリンク作成

Fortran95 ソースコード

```
!= Module module_name_mod : Sample module
! Authors:: Yasuhiro MORIKAWA
!
! This module depends base_mod module
!
module sample_mod
  use base_mod
  implicit none
  private
  public :: sample_init, sample_end, Const

  real(8) :: Const = 3.14

  subroutine sample_init(inchar, outint)
    character(*)    , intent(in) :: inchar
    integer(INTKIND), intent(out):: outint
  end subroutine sample_init

  subroutine sample_end(err)
    logical, intent(inout) :: err
  end subroutine sample_end

end module sample_mod
```

```
!= Module ..    ①
! Authors:: ..
```

```
use base_mod    ②
      :
module module_..
      :
  subroutine mo..
      :
  end subrou..
end module modu..
```

① RDoc のタグは “=“ や “::” で表記

② module 文, use 文, subroutine 文を解釈。別ファイルのモジュールへのリンクを自動で作成

Fortran95 Parser (オリジナル版)

- **Fortran90/95 の文法を解釈**
  - RDoc のタグは XML に比べとても簡潔
  - 別ファイル内のモジュール等へ自動的にリンク作成

Fortran95 ソースコード

```
!= Module module_name_mod : Sample module
! Authors:: Yasuhiro MORIKAWA
!
! This module depends base_mod module
!
module sample_mod
  use base_mod
  implicit none
  private
  public :: sample_init, sample_end, Const

  real(8) :: Const = 3.14

  subroutine sample_init(inchar, outint)
    character(*)    , intent(in) :: inchar
    integer(INTKIND), intent(out):: outint
  end subroutine sample_init

  subroutine sample_end(err)
    logical, intent(inout) :: err
  end subroutine sample_end

end module sample_mod
```

Rdoc

HTML のリファレンスマニュアル

Files	Classes	Methods
sample.f90 base.f90	sample_mod base_mod	sample_init sample_end

Class sample_mod
In: sample.f90

Module sample_mod : Sample module
Authors: Yasuhiro MORIKAWA

This module depends **base_mod** module

Methods
sample_init sample_end

Included Modules
base_mod

Public instance methods

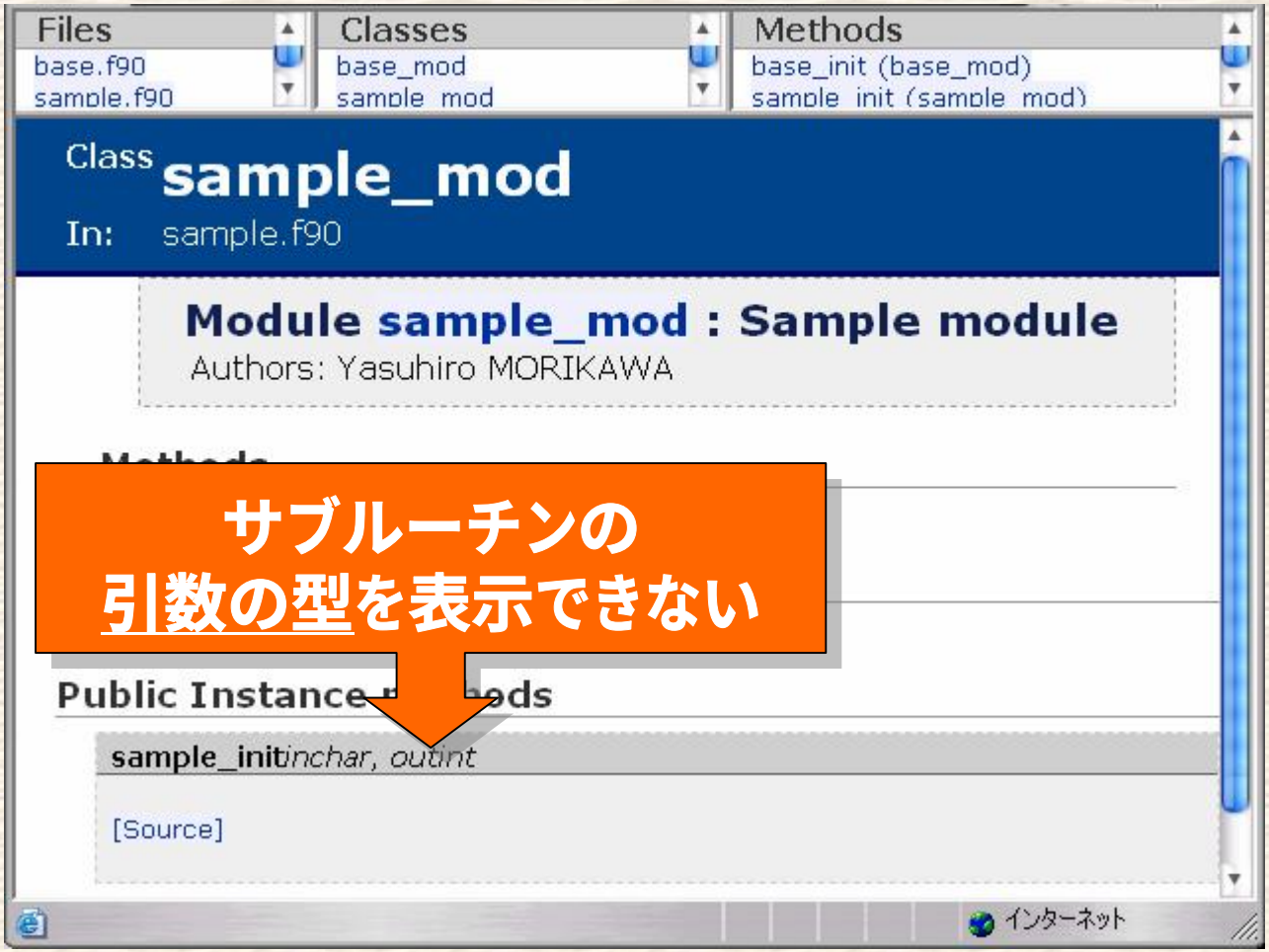
sample_init *(inchar, outint)*
[source]

sample_end *(err)*
[source]

ハイパーリンク

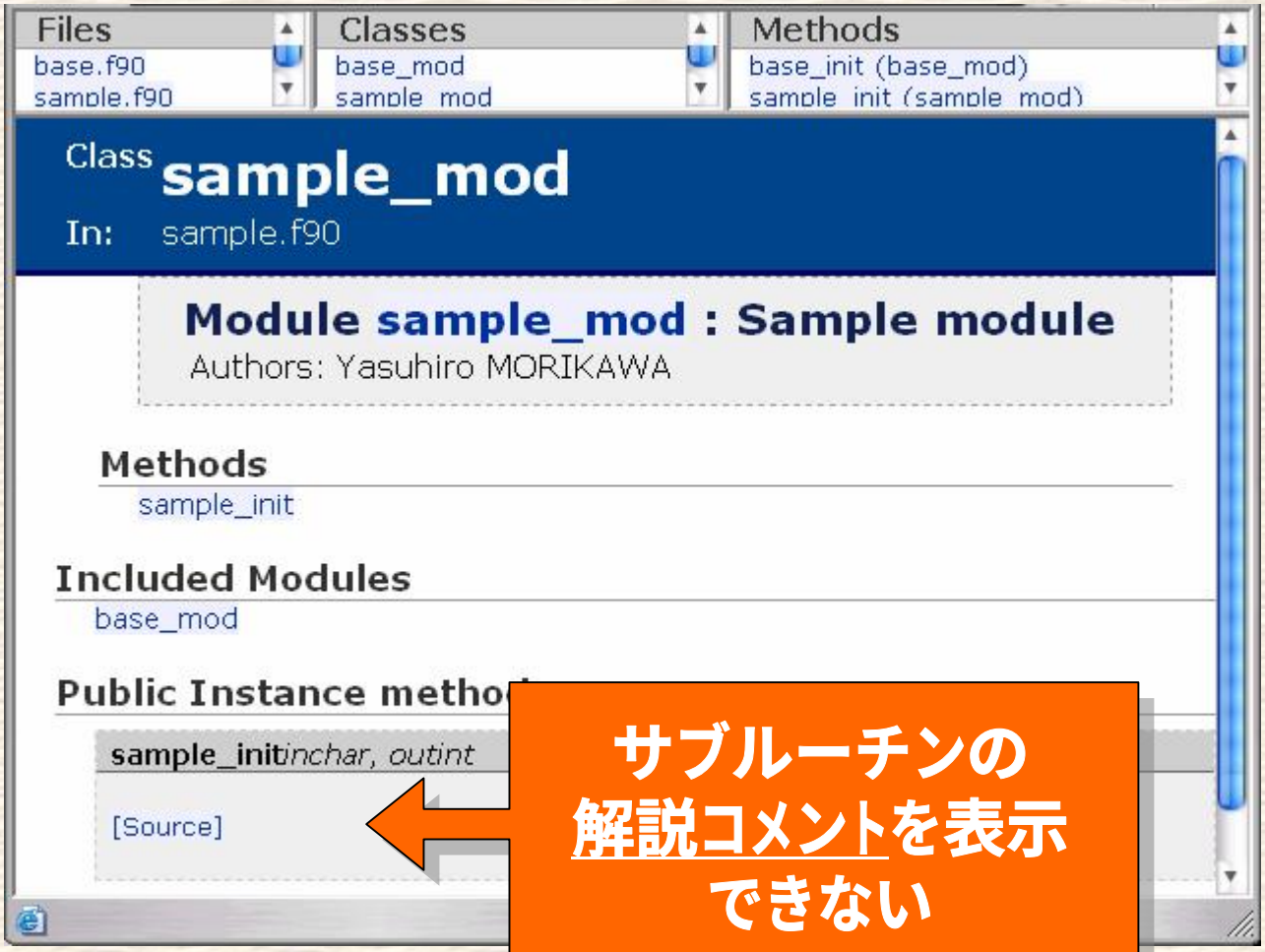
Fortran95 Parser の問題点

- 解析機能の不足



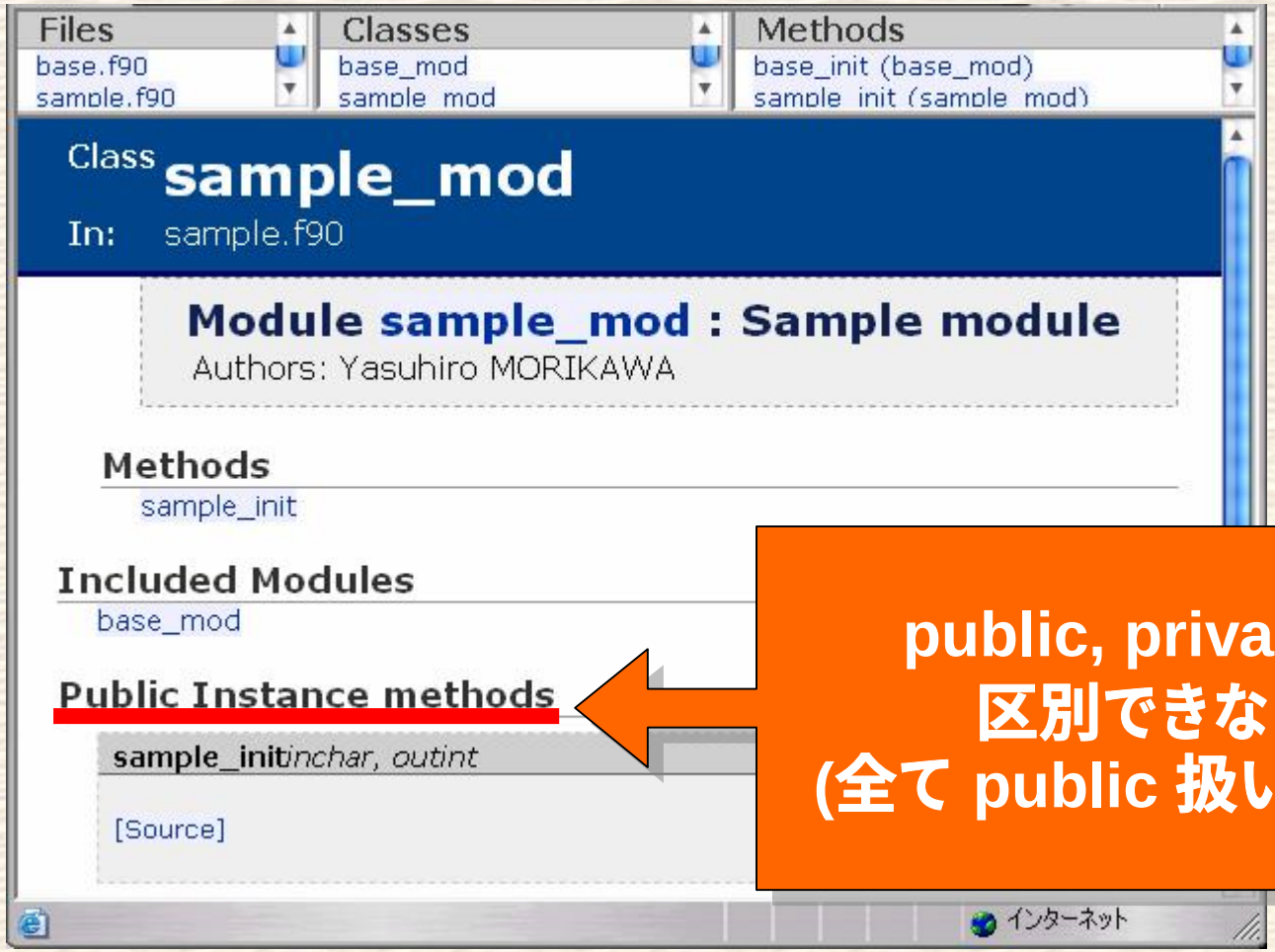
Fortran95 Parser の問題点

- 解析機能の不足



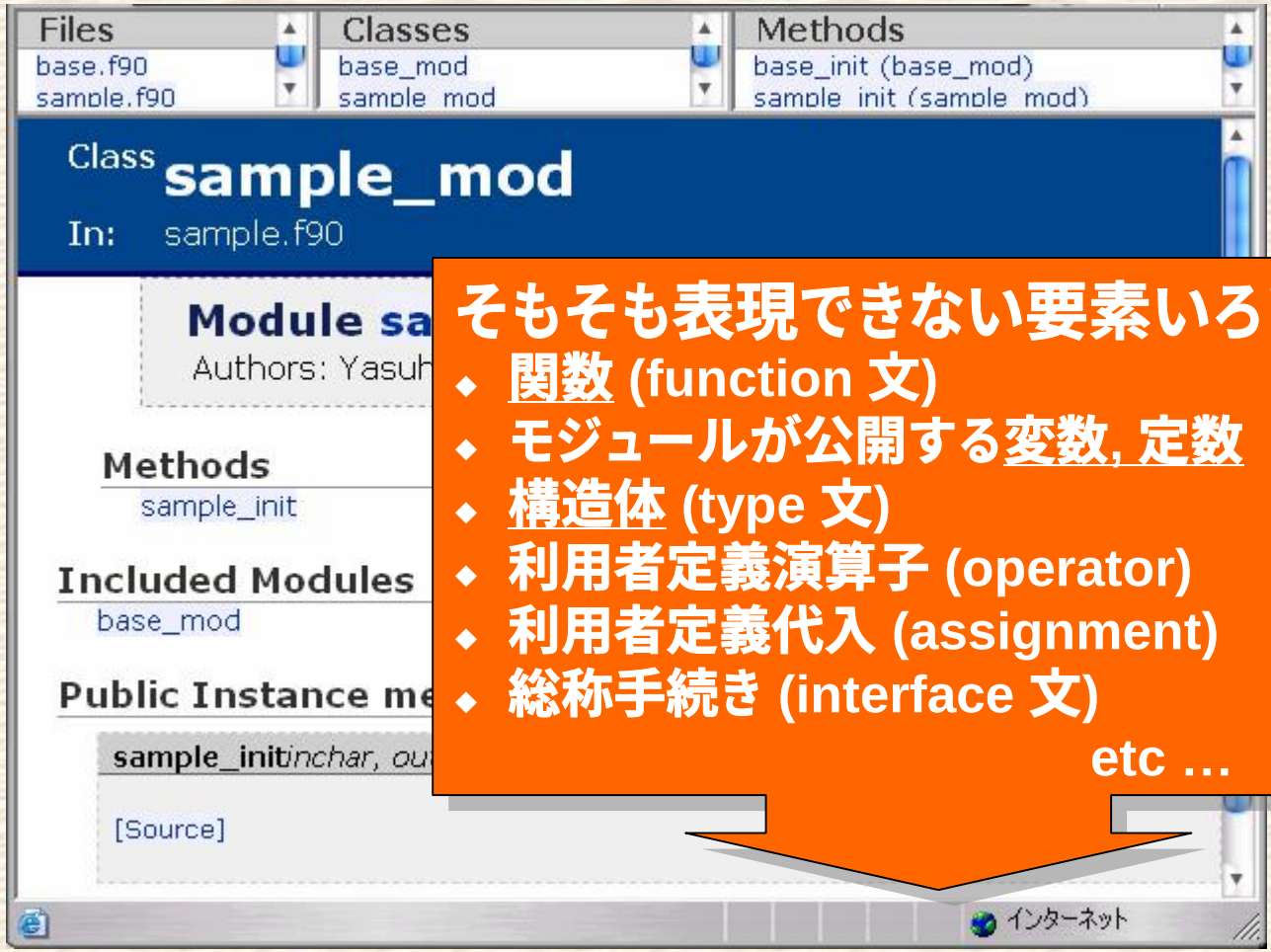
Fortran95 Parser の問題点

- 解析機能の不足



Fortran95 Parser の問題点

- 解析機能の不足



そもそも表現できない要素いろいろ

- 関数 (function 文)
- モジュールが公開する変数, 定数
- 構造体 (type 文)
- 利用者定義演算子 (operator)
- 利用者定義代入 (assignment)
- 総称手続き (interface 文)

etc …

問題点の解決に向けて

- 開発者･メンテナに連絡
  - 森川　　「こうなると嬉しいのだけど」と連絡
  - 開発者　「メンテナさんに連絡しておいたよ」
  - メンテナ「やってみてくれます?」
  - 森川　　(T_T)
- しょうがないので (?) 自力で改造
  - 見た感じで足りないと思うところから五月雨的に解析機能追加

Fortran95 Parser (強化版)

構造体の解析
要素、要素の型、解説の表示

Fortran95 Parser (強化版)

const()

Constant :

```
const = 3.14 : real(8), parameter
```

：公開定数

sample_fun

Function :

定数の解析
型、初期値、解説の表示

[Source]

sample_init(*inchar, outint*)

Subroutine :

```
inchar :      character(*) , intent(in)
```

：入力変数

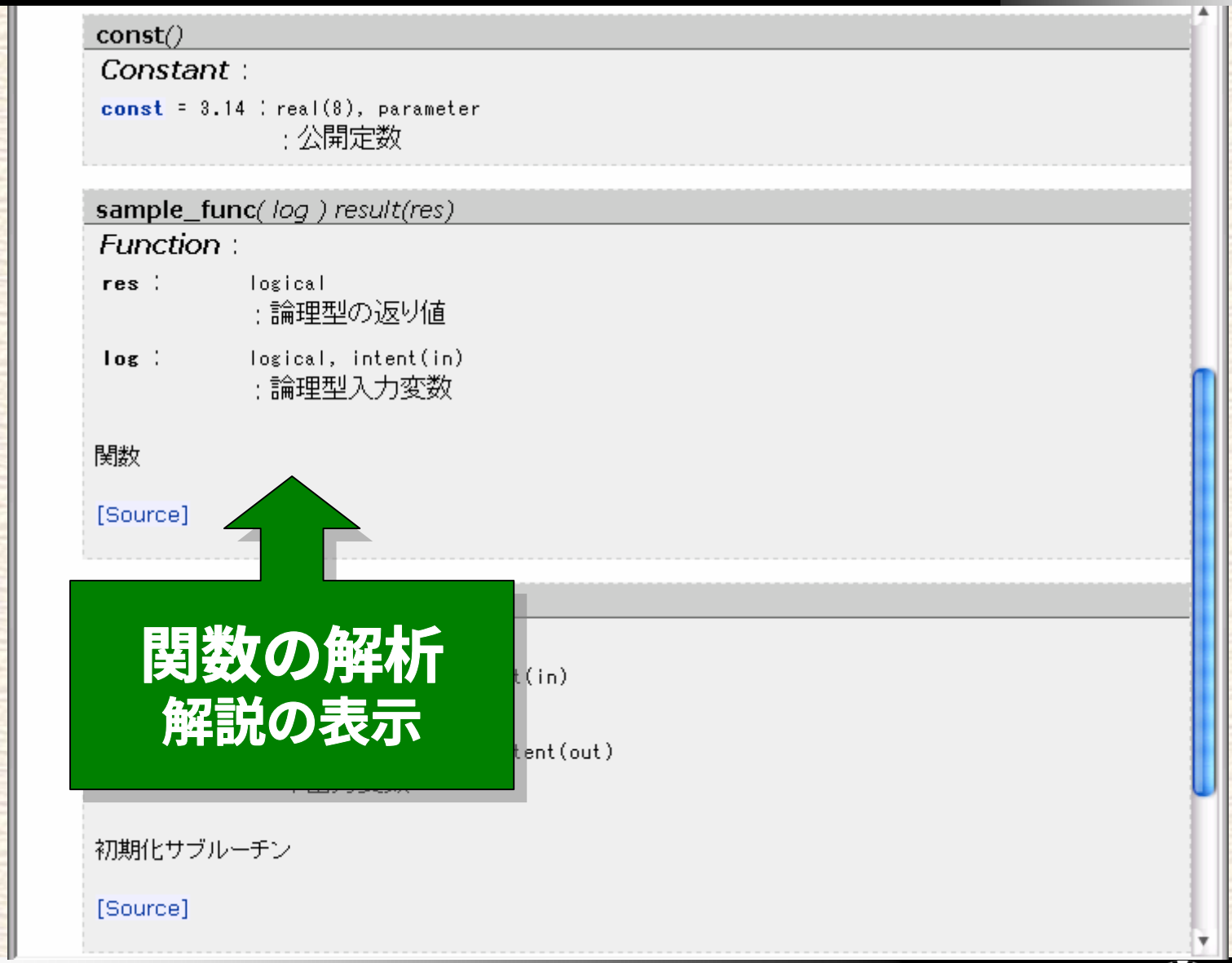
```
outint :      integer(INTKIND), intent(out)
```

：出力変数

初期化サブルーチン

[Source]

Fortran95 Parser (強化版)



関数の解析
解説の表示

Fortran95 Parser (強化版)

const()

Constant :

```
const = 3.14 : real(8), parameter
```

：公開定数

sample_func(*log*) result(res)

Function :

```
res :       logical
```

：論理型の返り値

```
log :       logical, intent(in)
```

：論理型入力変数

関数

[Source]

sample_init(*inchar, outint*)

Subroutine :

```
inchar :    character(*) , intent(in)
```

：入力変数

```
outint :    integer(INTKIND), intent(out)
```

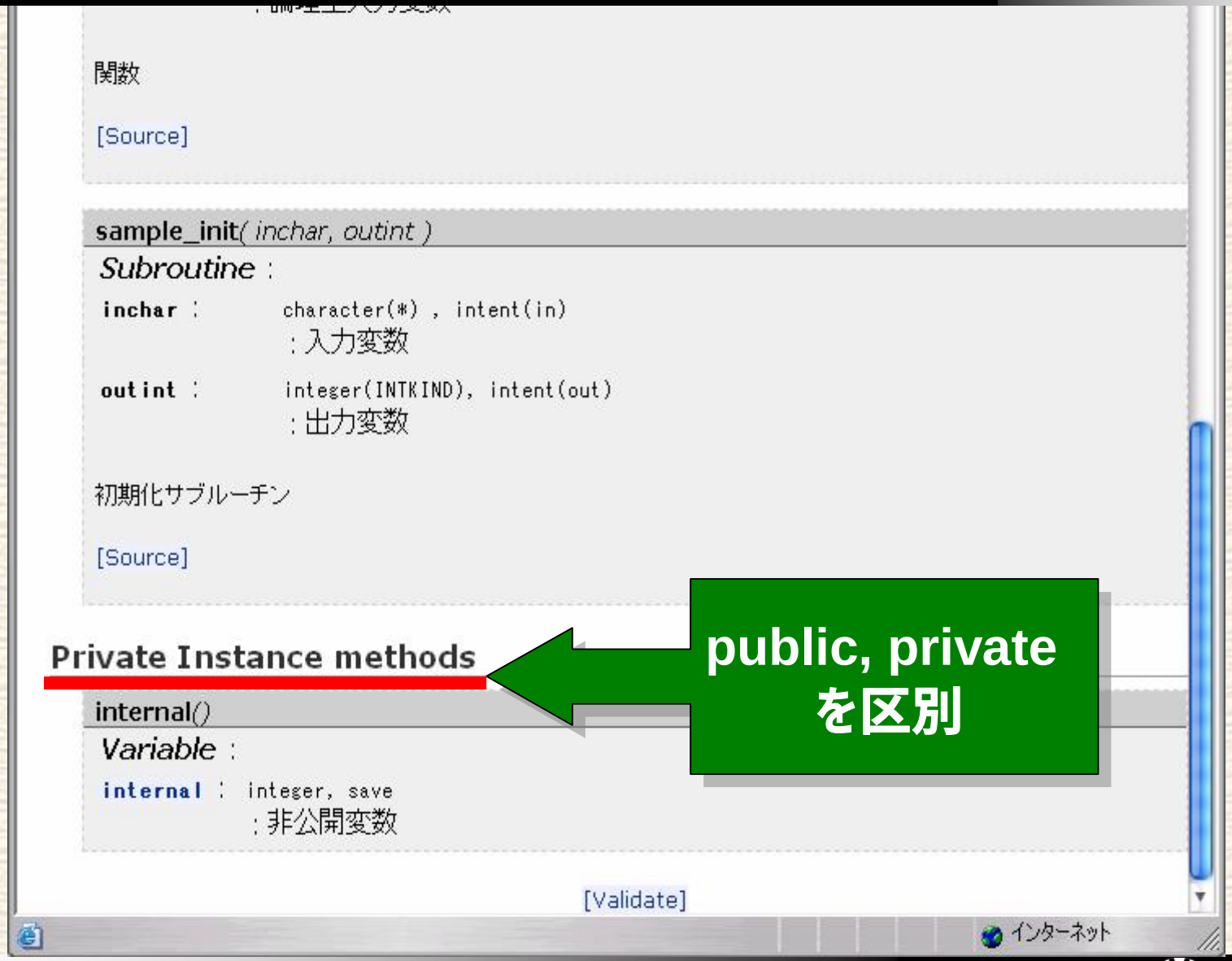
：出力変数

初期化サブルーチン

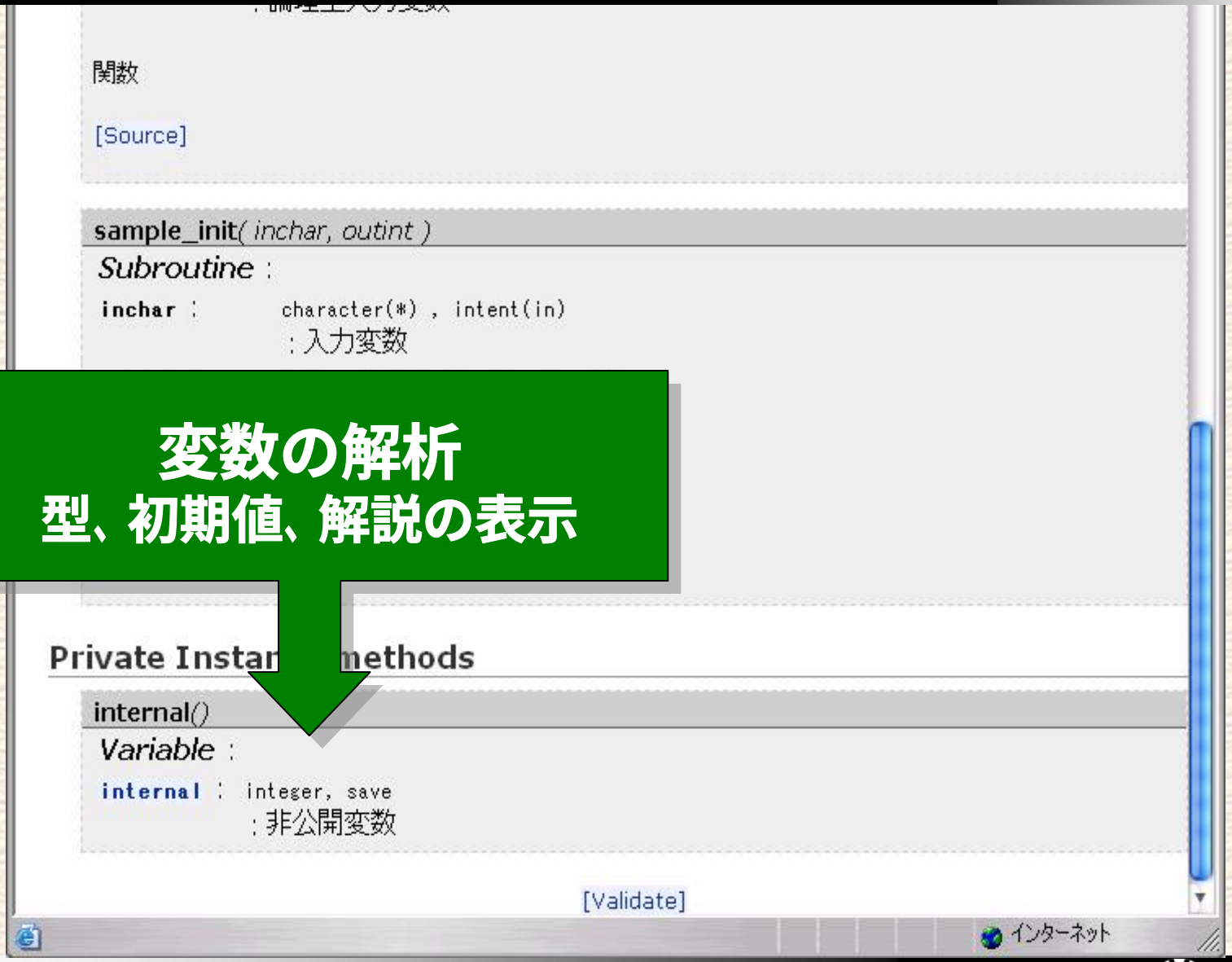
[Source]

引数の解析
型、解説の表示

Fortran95 Parser (強化版)



Fortran95 Parser (強化版)



変数の解析
型、初期値、解説の表示

まとめ

- RDoc の Fortran95 Parser を改良
  - 数値モデルのドキュメントを簡単に自動生成
    - モデルを渡す, もらう, 久しぶりに見直すときに是非 (?)
- 強化版パッチの公開アドレス
  - http://www.gfd-dennou.org/library/dcmodel
    - パッチを当てたパッケージ
    - コメントの書法等の解説
  - メンテナの依頼によりパッチを送付。そのうち Ruby 本体に取り込まれる予定 (たぶん)
- 使用例
  - http://www.gfd-dennou.org/library/gtool4

参考資料

- **数値モデリングプロジェクト dcmodel**
  - http://www.gfd-dennou.org/library/dcmodel/
- **オブジェクト指向スクリプト言語 Ruby**
  - http://www.ruby-lang.org
- **Fortran からドキュメントを自動生成するツール**
  - f90tohtml
    - http://mensch.org/f90tohtml/
  - f90doc
    - http://theory.lcs.mit.edu/~edemaine/f90doc/
- **惑星大気モデル DCPAM**
  - http://www.gfd-dennou.org/library/dcpam/
- **FMS (Flexible Modeling System)**
  - http://www.gfdl.noaa.gov/~fms/
- **The FMS Manual**
  - http://www.gfdl.noaa.gov/~vb/FMSManual/

付録

XML を用いた FMS (GFDL)の試み

- **Fortran95 に XML のドキュメントを埋め込む**
  - FMS (Flexible Modeling System: GFDL) **で試行**
  - FMS **特製のツールで** HTML **へ変換**

Fortran95 **ソースコード**

```
module module_name_mod
! <OVERVIEW>
!   module_name_mod の概要
! </OVERVIEW>
  implicit none
  private
  public :: module_name_init, module_name_end
  ! <SUBROUTINE NAME="module_name_init">
  ! <OVERVIEW>
  !   モジュールの初期化
  ! </OVERVIEW>
  ! <TEMPLATE>
  !   module_name_init (inchar, outint)
  ! </TEMPLATE>
  ! <IN NAME="inchar" TYPE="character" >
  !   文字型の入力変数
  ! </IN>
  ! <OUT NAME="outint" TYPE="integer" >
  !   整数型の出力変数
  ! </IN>
  subroutine module_name_init(inchar, outint)
    character(*)    , intent(in) :: inchar
    integer(INTKIND), intent(out) :: outint
  end subroutine module_name_init
  ! </SUBROUTINE>
end module module_name_mod
```

HTML **のリファレンスマニュアル**

Module module_name_mod

OVERVIEW

module_name_end の概要

PUBLIC INTERFACE

module_name_init:

モジュールの初期化

PUBLIC ROUTINES

a. module_name_init

call module_name_init (inchar, outint)

INPUT

inchar　文字型の入力変数

[character]

OUTPUT

outint　整数型の出力変数

[integer]

RD を用いた我々のこれまでの試み

- **Fortran95 に RD 形式のコメントを埋め込む**
  - rdtool によって RD を HTML に変換

Fortran95 ソースコード

```
                                                    !=begin
!= Module module_name_mod : Sample module
! * Developers: Yasuhiro Morikawa
!== Overview
!module_name_mod の概要

module module_name_mod                                      !=end
  implicit none

                                                           -begin
  !== Public Interface
  private
  public :: module_name_init, module_name_end

  !== Procedure Interface
  !=== Subroutine module_name_init : モジュールの初期化
  !NAMELIST を入力し、グローバル変数を allocate する。
  subroutine module_name_init(inchar, outint, inoutdata)
    !==== Input
    character(*)    , intent(in) :: inchar
    !==== Output
    integer(INTKIND), intent(out) :: outint
                                                    !=end
  end subroutine module_name_init
end module module_name_mod
```

rdtool

HTML のリファレンスマニュアル

Module module_name_mod : Sample module

- Developers: Morikawa Yasuhiro

Overview

module_name_mod の概要

Public Interface

```
private
public :: module_name_init, module_name_end    ! subroutines
```

Procedure Interface

Subroutine module_name_init : モジュールの初期化

NAMELIST を入力し、グローバル変数を allocate する。

```
subroutine module_name_init(inchar, outint, inoutdata, inoutdb)
```

Input

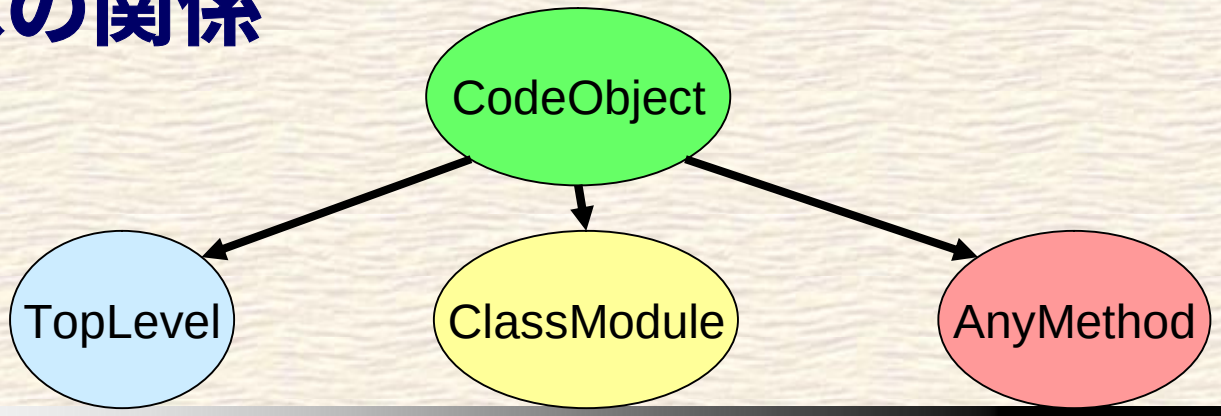
```
character(*)    , intent(in) :: inchar  ! Input Character
```

Output

```
integer(INTKIND), intent(out) :: outint ! Output Integer
```

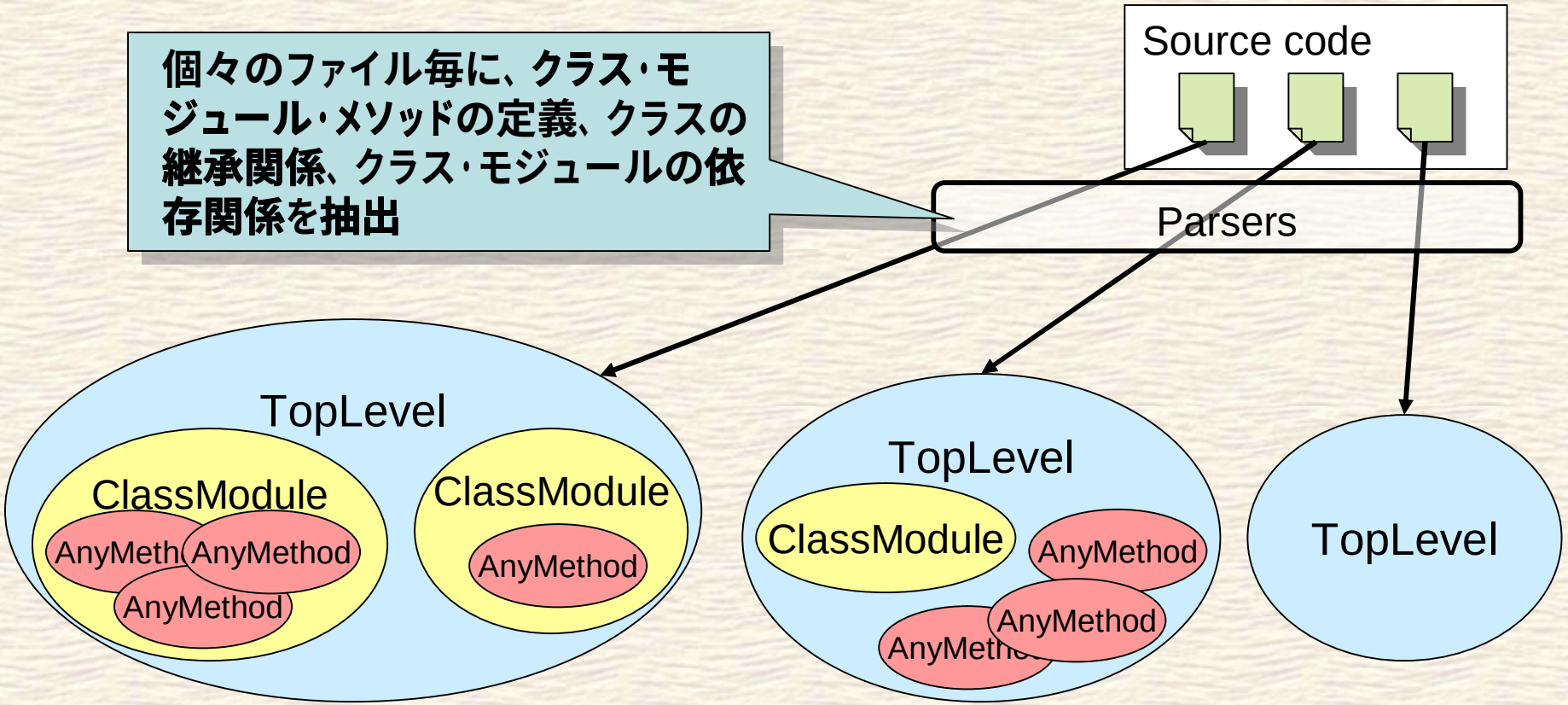
CodeObject

- ソースコード内の情報を階層的に保持するためのオブジェクト
  - CodeObject：共通する情報
  - TopLevel：ファイルの情報
  - ClassModule：クラス、モジュールの情報
  - AnyMethod：メソッドの情報
- 継承の関係



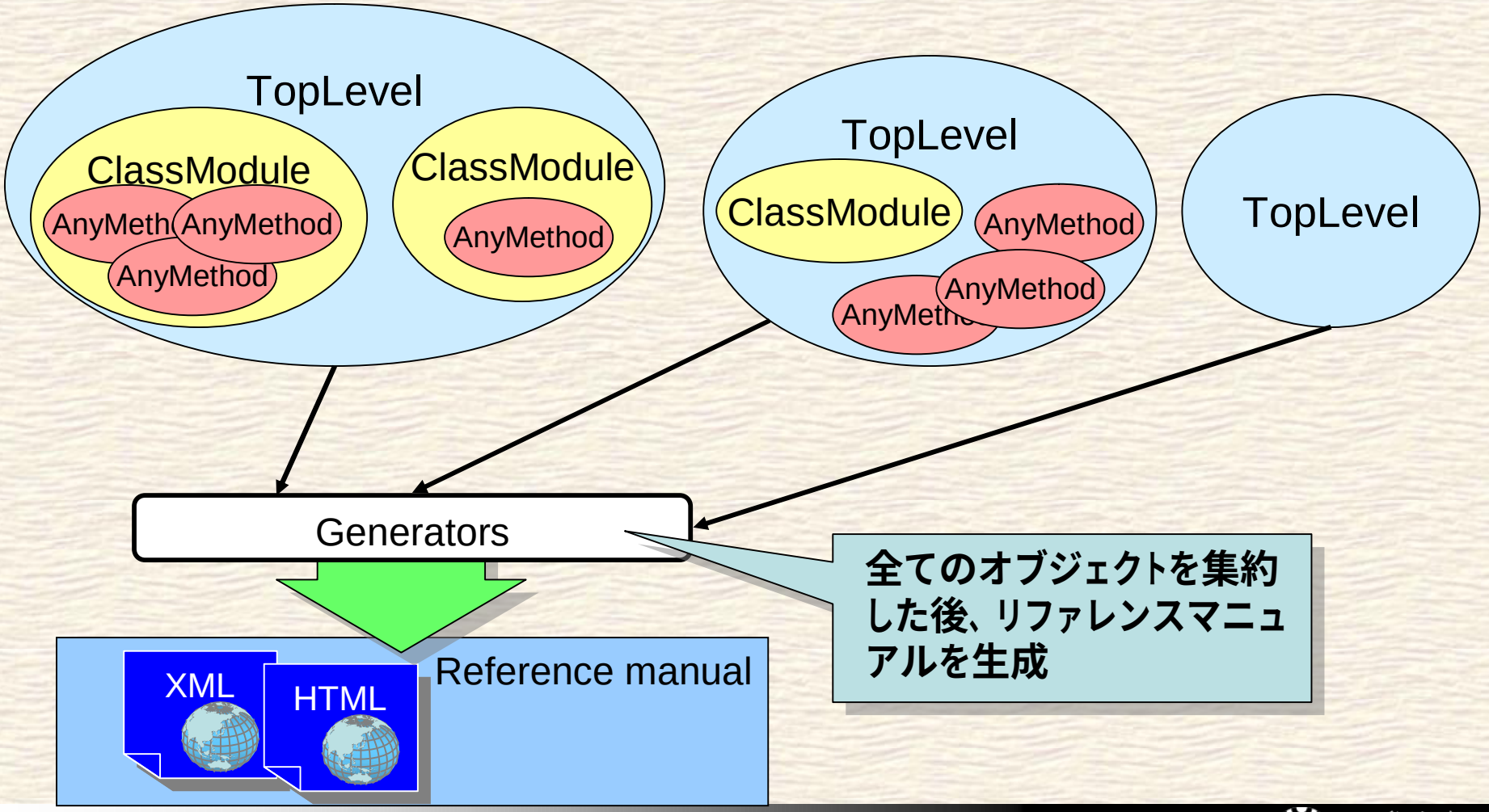
ソースコード ⇒ CodeObject

- ソースコード内の情報を TopLevel, ClassModule, AnyMethod へ



CodeObject ⇒ マニュアル

- HTML 等のドキュメントを生成



Fortran95 Parser (強化版)

■ 解析機能の強化の具体例

Fortran95 ソースコード

```
!= Module sample_mod : Sample module
! Authors:: Yasuhiro MORIKAWA
!
module sample_mod
  use base_mod
  !
  ! sample_mod の概要
  implicit none
  private
  public :: sample_init, sample_func, const, TYPE_A
  private:: internal

  type TYPE_A
    ! 構造体の解説
    integer :: counter    ! 構造体内部の変数の解説
  end type TYPE_A

  real(8), parameter :: const = 3.14   ! 公開定数
  integer, save      :: internal       ! 非公開変数

  subroutine sample_init(inchar, outint)
    ! 初期化サブルーチン
    character(*)     , intent(in) :: inchar  ! 入力変数
    integer(INTKIND), intent(out):: outint  ! 出力変数
  end subroutine sample_init

  function sample_func(log) result(res)
    ! 関数
    logical, intent(in) :: log  ! 論理型入力変数
    logical             :: res  ! 論理型の返り値
  end function sample_func
end module sample_mod
```

```
public:: samp...①
private:: inte...
```

```
type TYPE_A ...  ②
```

```
real(8), parame..③
Integer, save..
```

```
! 初期化...      ④
```

```
char..     ! 入力.⑤
integer..  ! 出力..
```

```
function samp...⑥
        :
end function ..
```

① 公開要素と非公開要素の区別

② 構造体

③ 公開定数、公開変数

④ サブルーチン、関数のコメント

⑤ 引数の型、コメント

⑥ 関数

Fortran95 Parser (強化版)

- **解析機能の強化**
  - 解析可能になった要素のリスト
    - 関数 (function 文)
    - サブルーチンや関数の引数の型
    - モジュールが公開する変数, 定数
    - 構造体 (type 文)
    - NAMELIST 文
    - 利用者定義演算子 (operator), 利用者定義代入 (assignment)
    - 上記要素のコメント文
    - 総称手続き (interface 文)
    - 公開要素と非公開要素との区別
    - 孫引きされている公開要素
  - 大文字小文字の違いを無視

RD を用いた我々のこれまでの試み

- **Fortran95 に RD 形式のコメントを埋め込む**
  - RD を用いることでタグが簡潔に
  - rdtool によって RD を HTML に変換

Fortran95 ソースコード

```
                                                    !=begin
!= Module module_name_mod : Sample module
! * Developers: Yasuhiro Morikawa
!== Overview
!module_name_mod の概要

                                                    !=end
module module_name_mod
  implicit none

                                                    !=begin
  !== Public Interface
  private
  public :: module_name_init, module_name_end

  !== Procedure Interface
  !=== Subroutine module_name_init : モジュールの初期化
  !NAMELIST を入力し、グローバル変数を allocate する。
  subroutine module_name_init(inchar, outint, inoutdata)
    !==== Input
    character(*)    , intent(in) :: inchar
    !==== Output
    integer(INTKIND), intent(out) :: outint
                                                    !=end
  end subroutine module_name_init
end module module_name_mod
```

```
!=begin      ①
   :
!=end
```

```
!= Module ..②
! * Develop..
!== Overview
```

```
 ③      !=begin
   :
subroutine ..
  char.., int..
   :
      !=end
```

①!=begin 〜 !=end の部分をマニュアルに反映

②コメントには RD の文法を利用。見出しやリストは “=” や “*” で簡潔に表現

③引用仕様 (引数の名前と型など) に関するソースコードをそのままマニュアルへ

RD を用いた我々のこれまでの試み

- Fortran95 に RD 形式のコメントを埋め込む
  - 利点
    - ソースとマニュアルとで 2 度書く手間を軽減
    - XMLタグと比較してソースが汚れない
  - 欠点
    - モジュール間の依存関係を解釈不能
      - 複数のファイルを同時に解析しないと不可能
    - ソース内のタグ (“=begin” 等) はやっぱりそれなりに汚い

- ソースコード自動解析の必要性

RDoc によるドキュメント生成の流れ