Fortran における UnitTest 環境

佐々木洋平/Youhei SASAKI

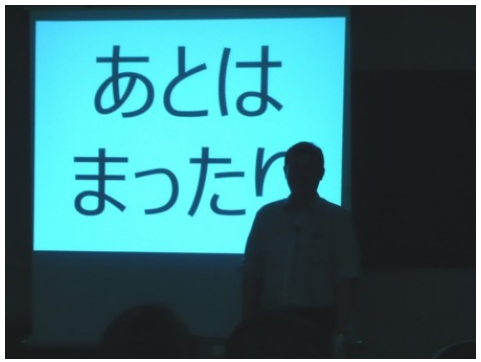
2012 年 3 月 5 日

Outline

Outline

About me...

- 名前: 佐々木洋平
- mail: uwabami@gfd-dennou.org
- twitter/IRC: uwabami
- 所属:
  - 京大・数学教室
  - 地球流体電脳倶楽部
  - Debian Project/Debian JP Project
    - Science, Ruby/Extras, GIS, TEX,...



- 専門: 回転球殻対流, MHD ダイナモ (主に数値計算)
- 趣味: 計算機いじり, Debian メンテ (主に Ruby, TEX ...)

今日のお話

Fortran(>=90) 用の UnitTest フレームワークを作成

- Fortran で閉じている→可搬性, 移植性
- 高度な Assertion: 森川, 他 (2010) のさらなる改良
- Free Software

今日のお話

Fortran(>=90) 用の UnitTest フレームワークを作成

- Fortran で閉じている→可搬性, 移植性
- 高度な Assertion: 森川, 他 (2010) のさらなる改良
- Free Software

- しかし Ruby の人々に対して UnitTest って...
  - 今更ですよねー.
  - しかし, 電脳倶楽部の人々の多くは UnitTest を知りません
  - そんな訳で西澤さんに呼ばれて喋ります.

Outline

ソフトウェアのテストと開発スタイル

- ソフトウェアの「テスト」
  - ソフトウェアプログラムが正しく動作することを確認する作業.
  - 仕様の検証, 欠陥の発見, バグ潰し...
- Test-Driven Development(テスト駆動開発, TDD)
  - テスト主導の開発スタイルの名称
  - TDD の流れ
    1. 現在の仕様を確認して, テストの必要がある項目を列挙する
    2. 上記項目に合わせてテストを書く→失敗することを確認する
    3. テストが成功する最小限のコード本体を実装する
    4. リファクタリング

テスト?

- Unit Test(単体テスト)
  - メソッドなどの小さな単位で行なうテスト
  - fortran → `function`, `subroutine` 単位でのテスト
- Integration Test(結合テスト)
  - Unit Test が完了したプログラムを組み合わせて行なうテスト
  - 数値モデル→実際に計算した結果の検証

現実は?

- 多くは「ビッグバンテスト」
  - 全体が完成してから行なわれるテスト
  - 規模が大きくなると問題の原因の追求が困難!!
- 数値モデルでは?
  - 全体を「結合」するために, 多くの修正が追加される
    - UnitTest が無い→正しく「変形」したのかがわからない
- とはいえ数値モデルの「結合テスト」は面倒だが
  - 何が「正しい答」がわからない (事が多い)
    - ここではこれ以上触れない

Outline

xUnit

- xUnit = Unit Test フレームワークの総称
- xUnit の設計
  - テストフィクスチャ
    - 必要な数値や条件の集合
  - テストスイート
    - 同じフィクスチャを共有するテストの集合
  - アサーション
    - テスト対象の関数, クラスについて検証を行なうための関数, マクロ
  - テストドライバ
    - テストを実行するプログラム.

xUnit の例: fortran90 + dc_test

```
use dc_test
use hoge_module
!= load fixture
!  テスト実行に必要な設定(フィクスチャ)の読み込み
call dc_test_setup(hoge_module_fixture)
...
! hoge_module の fuga の assertion
call assert_equal(                          &
     message = 'test of fuga(input)', &
     answer  = fuga_answer,                 &
     check   = fuga(input),                 &
)
...
!= teadown
!  テスト終了後のフィクスチャテのクリア
call dc_test_teardown
```

Fortran での Unit Test フレームワークあれこれ

- 既存の UnitTest フレームワーク
  - FUnit: http://nasarb.rubyforge.org/funit/
    - Fortran(>= 77), ドライバとフィクスチャ等は Ruby で実装. NASA OpenSource Agreement(non-FSF approval)
  - FRUIT: http://fortranxunit.sourceforge.net/
    - Fortran(>= 95), ドライバとフィクスチャは Ruby で実装. License 不明
  - pFUNIT: http://sourceforge.net/projects/pfunit/
    - Fortran(>= 90). Pure Fortran. (C)US Gov. and NASA (PublicDomain?)
  - Lutin77: http://www.logilab.org/project/lutin77
    - FORTRAN77 用. ドライバは C で実装. GPL
  - ObjexxFTK: ドライバは Python. 売り物. 良く知らない

既存の Fortran UnitTest の気に入らない所

- Assertion がショボい
  - AssertEqual が単精度のみってどういうこと?
  - Array まわりが貧弱だったり
  - pointer, interface に対する Assertion が無い
- ドライバが他の言語
  - Fortran のテストをするのに Ruby が必要って...
  - とはいえドコまで許容するのかな?
- ライセンス
  - Free じゃない.

Outline

dc_test の紹介

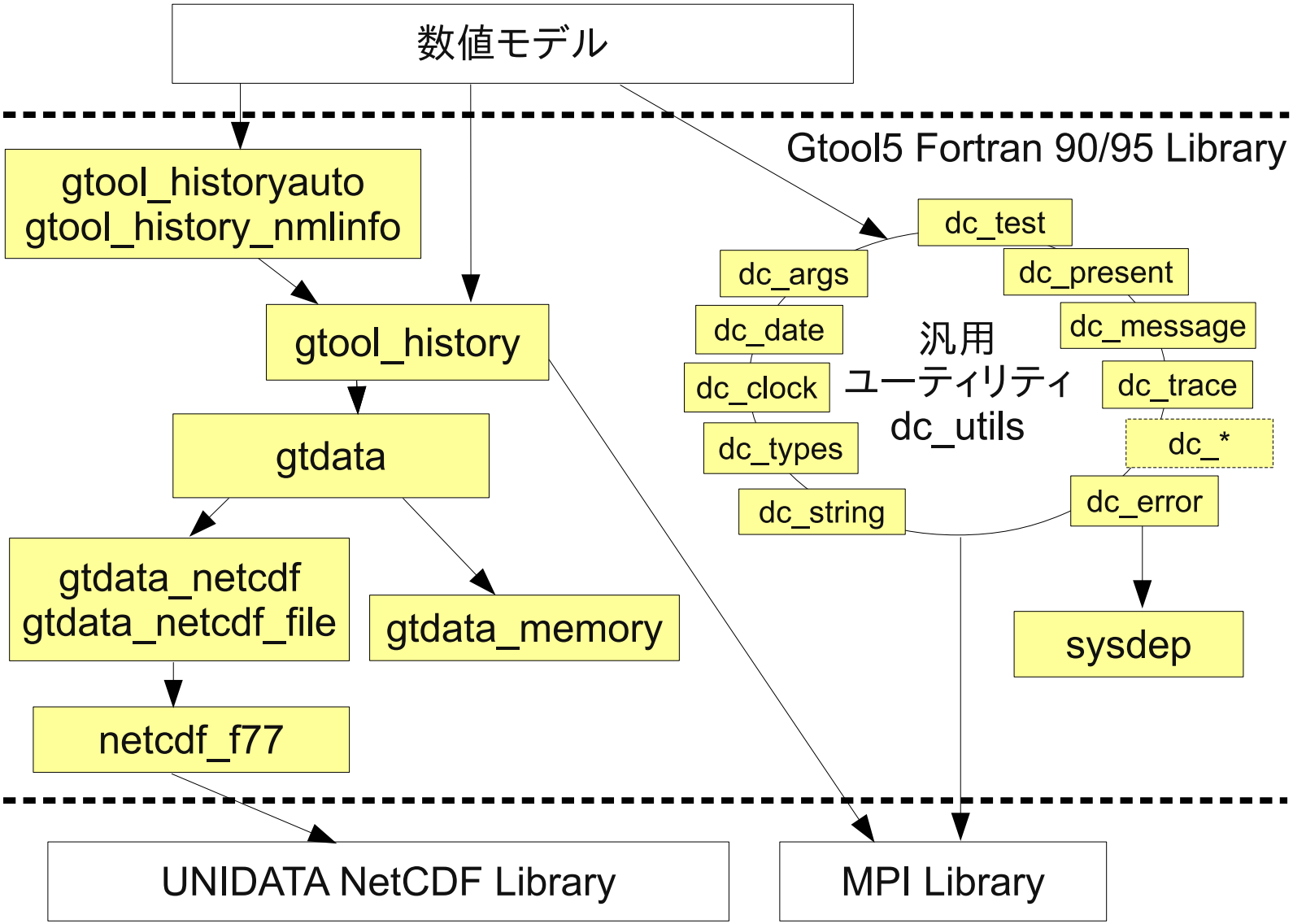
dc_test: Fortran(>=90) 用の UnitTest 用ライブラリ

- 豊富な Assertion: 倍精度, 4 倍精度, Pointer, Interface...
- ドライバも Fortran で記述
- Free Software: MIT ライセンス

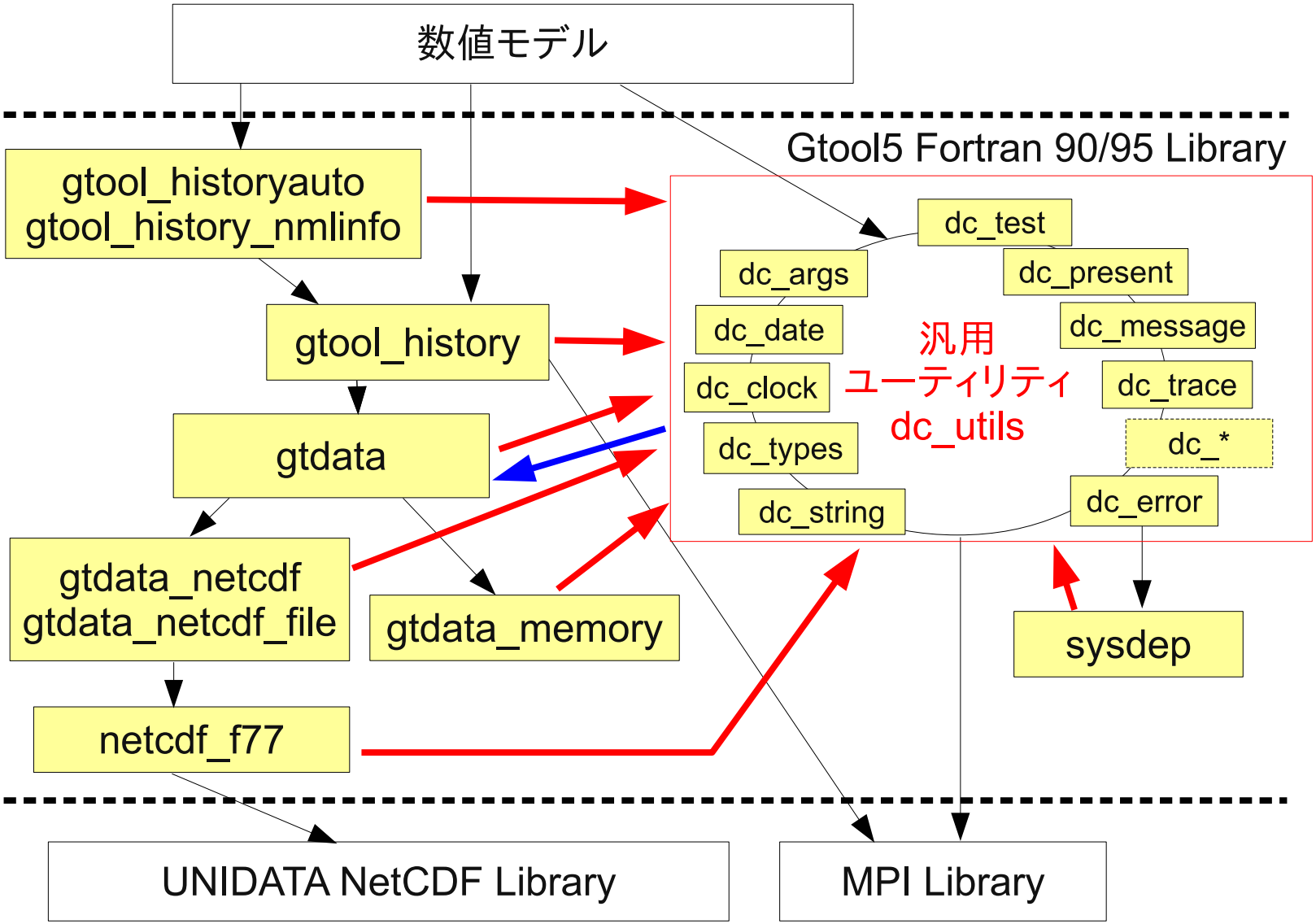
歴史 (1): 元々は...

- 地球流体電脳倶楽部 gtool5 Fortran 90/95 ライブラリの一部
  - 森川靖大, 他 (2010): http://www.gfd-dennou.org/library/gtool/
    - 多次元格子点データのための自己記述的データ I/O と数値モデルのための汎用ユーティリティ
  - 2010 年から佐々木が引き継いで開発継続, 管理

歴史(2): gtool5 の設計



歴史(3): gtool5 の設計



歴史 (4): 元々は...

- 地球流体電脳倶楽部 gtool5 Fortran 90/95 ライブラリの一部
  - 森川靖大, 他 (2010): http://www.gfd-dennou.org/library/gtool/
    - 多次元格子点データのための自己記述的データ I/O と数値モデルのための汎用ユーティリティ
  - 2010 年から佐々木が引き継いで開発継続, 管理
- gtool5 ライブラリから dc_utils を分離
  - 整理, 汎用性の向上...
  - dc_test: フィクスチャ管理とドライバの追加
    - dc_test_setup, dc_test_teardown...

Outline

SPMODEL, spml

- SPMODEL: 階層的地球流体スペクトルモデル集
  - 地球流体力学に登場するさまざまなレベルの近似方程式系の数値モデルを, 空間 1 次元モデルから 2 次元あるいは 3 次元モデルまで階層的に整備
  - http://www.gfd-dennou.org/library/spmodel/index.htm

spml: スペクトル計算用の Fortran95 ライブラリ

「数式のようにプログラミング」

- Fortran(>=90) の配列演算機能により
- 各種境界条件に対応した微分演算を提供

spml の例: ae_module の使用例

一次元移流方程式を ae_module で解く場合

$$\frac{\partial \zeta}{\partial t} = -c\frac{\partial \zeta}{\partial x} \rightarrow \zeta(t+\Delta t) = \zeta(t) + \Delta t * \{-c * \partial_x \zeta(t)\}$$

```
use ae_module
...
! スペクトル初期化
call ae_initial(im,km,xmin,xmax)
...
! Euler scheme 時間積分
do it=1,nt
   e_Zeta = e_Zeta + dt * ( -c * e_Dx_e(e_Zeta) )
   ...
enddo
...
```

spml の例: ae_module が提供する関数など

- 初期化: ae_initial
  - 座標変数: g_X
  - 重み座標: g_X_Weight
- 基本変換: g_e, ag_ae, e_g, ae_ag
- 微分: e_Dx_e, ae_Dx_ae
- 積分: Int_g, a_Int_ag, Avr_g, a_Avr_ag
- 補間: Interpolate_e, a_Interpolate_ae

spmlでの使用例: ae_initial_test.f90

```
use dc_test
use ae_module
!= load fixture
call dc_test_setup(ae_module_fixture)

!= 実行
call ae_initial(im, km, xmin, xmax)
...
!= assertion of g_X_Weight
call assert_equal(                          &
     message = 'test of g_X_Weight',        &
     answer  = g_X_Weight_answer,           &
     check   = g_X_Weight                   &
)
...
!= teadown
!  テスト終了後のフィクスチャのクリア
call dc_test_teardown
```

まとめ

Fortran(>=90) 用の UnitTest フレームワークを作成

- Fortran で閉じている→可搬性, 移植性
- 高度な Assertion: 森川, 他 (2010) のさらなる改良
- Free Software

課題

- ドキュメント (チュートリアル, サンプル) の作成, 公開
- ジェネレータをどうするか?
  - そもそも提供しない, という手もあるけれど...

参考文献

- FUnit: http://nasarb.rubyforge.org/funit/
- FRUIT: http://fortranxunit.sourceforge.net/
- pFUNIT: http://sourceforge.net/projects/pfunit/
- Lutin77: http://www.logilab.org/project/lutin77
- gtool5, 森川靖大, 他 (2010): http://www.gfd-dennou.org/library/gtool/
- spml, 竹広, 他 (2011): http://www.gfd-dennou.org/library/spmodel/index.htm