RELAP5-3D Conversion to Fortran 90

RELAP5 International Users Seminar

Dr. George L. Mesina

September 7-9 , 2005

Idaho National Laboratory

Outline

- Purpose
- High-level Task Description
- Progress
- Conversion methodology
- Measurements

What Users Want from Conversion

- Don’t change my answers!
  - Horror stories from other code conversions
  - Elaborate testing prevents changes
- Don’t cripple my feature!
  - Parallel, vector, PVM, restart, GUI, PYGI, etc.
- Improve RELAP5-3D somehow.

Improvements from Conversion

- Increase machine independence.
  - Use Fortran 90 intrinsics, not MILSPEC.
- Eliminate memory restrictions.
  - Replace FA-array with allocatable arrays.
- Longevity.
  - Replace any Fortran 66 & 77 constructs that may be illegal in future versions of Fortran.
- Modernization.
  - Convert to derived types (proto-object-oriented).
  - Use whole array operations.

Improvements from Conversion

- Coding and data structure simplified for readability and understandability
  - Less time required for code development
  - Reduced debugging time
  - Reduced cost for maintenance
- New developers will learn the code faster
  - More modern language, programming constructs, and programming style

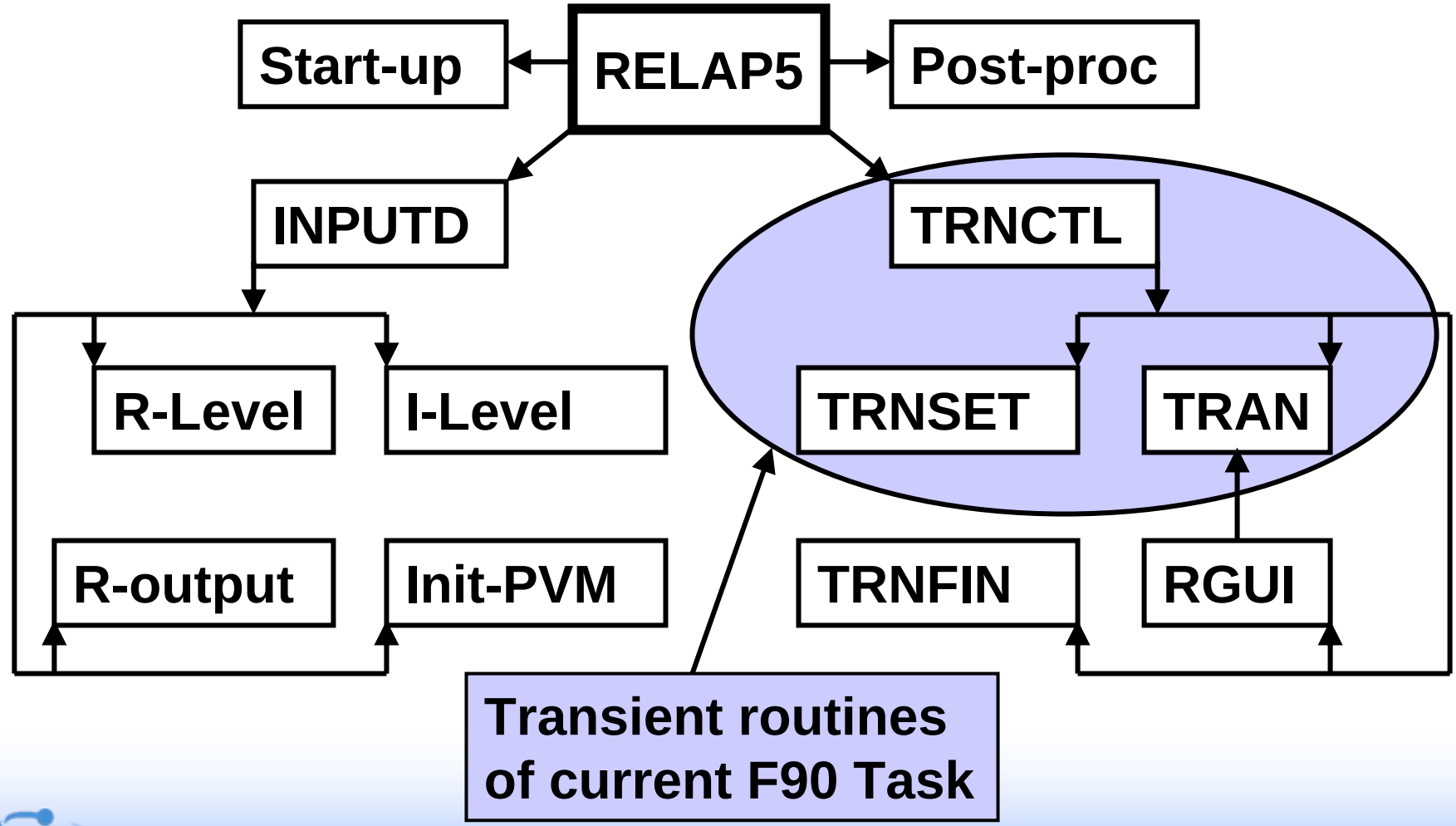
Specific Goals of Fortran 90 Task

- Replace obsolete coding structures.
- Replace COMDECKS with Fortran 90 Modules.
- Replace “internal FA-Files” with derived types.
- Simplify the labyrinthine data structure.
  - Replace index variables with ordinals.
  - Replace LOCF and indexing-pointer method with Fortran 90 (real) pointers.
- Replace MACH* with machine-independent, Fortran 90 intrinsic functions.
- Ultimately, eliminate FA array and FTB.

High Level Description: Order

- Order of conversion
  - By functional groupings
    1. Transient routines
    2. I/O routines
    3. Environmental routines
    4. Others
  - By internal FA Files with a functional group
    - 47 of them
  - By “calling trees” within an FA file in leaf to root order.

High Level Description: Function



High Level Description: FA Categories

- **Categories of FA files.**
  1. ***Standard*** **– Single fixed stride through memory**
  2. ***Interwoven*** **– 2+ fixed strides through memory**
  3. ***Complex*** **– some characteristics are:**
     - **Non-fixed strides**
     - **Referenced by direct FA access,**
     - **Multiple equivalence for single FA index**
  4. ***Removable*** **– No longer used.**

High Level Description: Conversion

- To convert an FA file
  1. Develop conversion tools to automate conversion.
  2. Create module.
  3. Convert all subroutines that use the FA file

     3a. For a given subroutine
     - Convert subroutine with tools.
     - Test that code performance is unaffected.
     - Make manual modifications as needed.

     - Repeat Step 3a until all subroutines of the FA file are successfully converted.

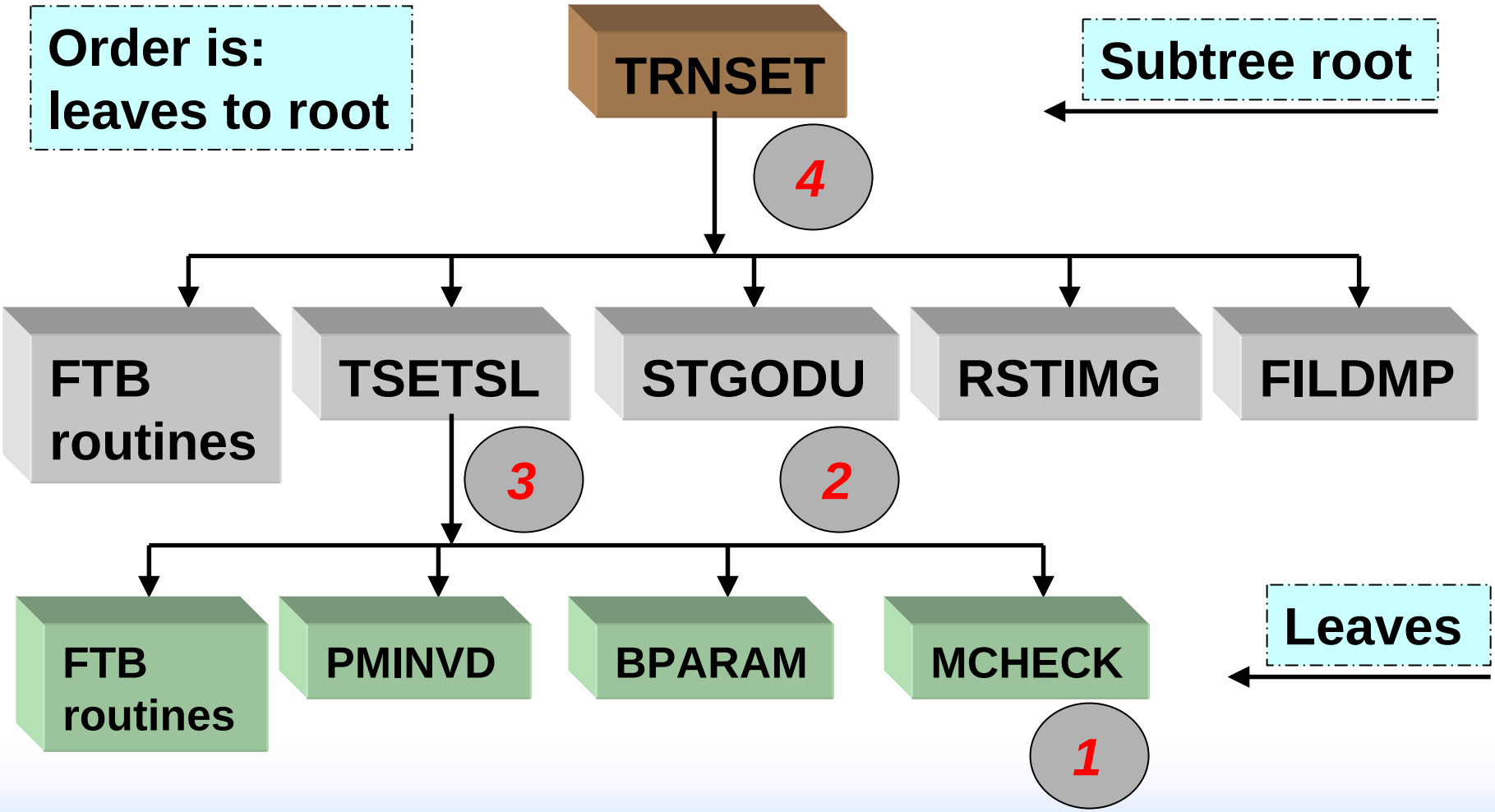
Progress (as of Version 2.5.1)

3 standard (category 1) FA files have been fully converted.

FA file	Module created			Converted	
	Name	Subpgms	# lines	Subpgms	# lines
VOLDAT	VOLMOD	5	1093	158	12216
JUNDAT	JUNMOD	5	721	101	7220
LPDAT	LPDMOD	5	321	96	2407
Total *		15	2135	355 *	19833

* Each separate conversion of a subroutine is counted.

Conversion Order of an FA File Subtree



Reason for "Leaves to Root" Order

- **When converting the FIRST subroutine, all the most-current data is in the FA-array.**
- **Must *upload* (copy) FA-data *to module*.**
  - **Upload performed first action at top of routine.**
- **Calculations in converted subroutine are performed in derived type arrays.**
  - **The most-current data is now in the module.**
- **Must *download* module data *to FA-array*, last action before returning.**
- ***What would happen if the converted subroutine called an unconverted subroutine?***

Some Conversion Rules

1. **No subroutine can be converted until all the subroutines it calls are converted.**
2. **Subroutines called from a *converted subroutine* must NOT perform data uploads or downloads.**
   - **What would happen if one did an upload from FA to module?**
3. **Control data transfers from the calling routine.**
   - **Important for subroutines called from both *converted* and *unconverted* subroutines.**

Method to Convert a Subroutine

1. Pre-process with conversion tools
2. Convert to derived types with conversion tools
3. Post-process the converted file
   - Fix compiler errors
   - Run small test set
   - Debug runtime errors
   - Debug differences in calculations

Pre-processing a Subroutine

1. Place the “use module” statement.
2. Place the transfer statements at start and return.
   - Upload (start), download (return), and controls.
3. Create ordinals for indexing derived type arrays.
4. Split indices of multi-D, FA-file arrays in two.

   dl(ivp1) = dl(iv) BECOMES dl(iv+1) = dl(iv)
5. Turn array references in do-loop limits into scalars.
6. Declare and create assignment statements for the variables in 3, 4, and 5.
7. Apply some automated RELAP5 style rules.

“Derived Type Processing” a Subroutine

- “Single-index” arrays, such as pressure, become derived type scalar attributes.

  p(iv) ==> vlm(miv)%p

  – Use ordinal index, miv, NOT FA-array index, iv.
- Convert multi-index arrays to derived type attribute vectors.

  dl(ivp1) ==> dl(iv+1) ==> vlm(miv)%dl(2)
- Derived types extend statement length, stay within column 72 via continuation.

Post-processing a Subroutine

- Common compiler errors
  - Undeclared, newly created variables
  - Arrays with array subscripts are mishandled by the converter.
- Some runtime errors
  - A new variable created in an if-branch is undefined in the else-branch.
  - Multiple returns. Convert to single exit point.
  - Failure to split an index (that needs splitting) prior to conversion.

Testing

- Tested on small set of standard problems first.
  - Make sure it still does what it is supposed to do.
  - Check that it does not cause failures elsewhere.
- Tested on all “normal test problems”
  - Whenever a small subtree is completed.
  - At least once every 10 conversions.
- FA-file is considered FULLY CONVERTED when
  1. All test cases produce identical output to the unconverted code.
  2. “All” its transfers have been commented out.

Transient Conversion Information

- 47 internal FA-files
  - 1108 = Number of includes in all subroutines
  - 849 = Number of includes in all transient subs
- Fully converted FA-files
  - 158 = # includes of VOLDAT by transient subs
  - 101 = # includes of JUNDAT by transient subs
  - 96 = # includes of LPDAT by transient subs
- Progress on “Transient FA-conversion” task
  - 42% complete = (158+101+96) / 849