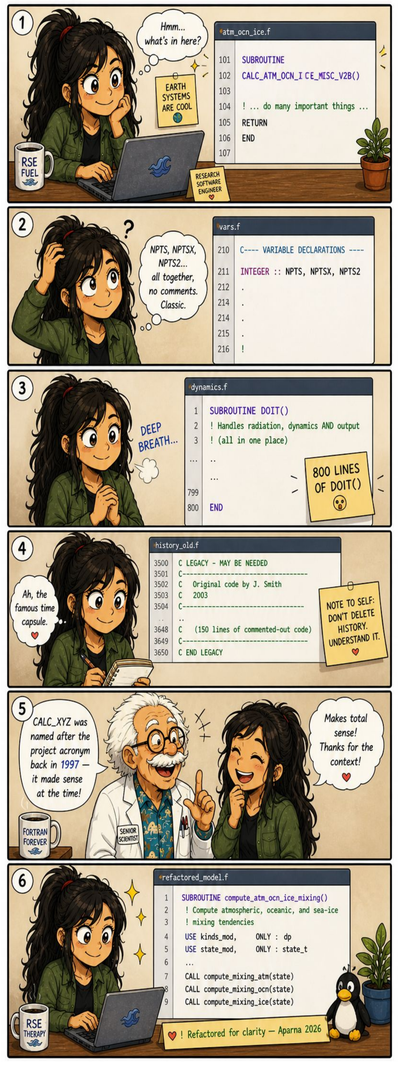
Navigating Legacy Earth System Model Software

An inside perspective from a Research Software Engineer | EGU 2026.ESSI3.4

Lakshmi Aparna ***Devulapalli****, Deutsches Klimarechenzentrum, Hamburg, Germany*



A day in the life of an RSE*

*Comic strip generated by AI

BLUF (Bottom Line up front) Many German ESMs originate in the 1990s, written predominantly in Fortran. While the broader scientific software world has moved toward Python and C/C++, ESM codes are still catching up. Research Software Engineers within projects like natESM must bridge this multi-decade gap — delivering modern, reproducible science workflows from codebases that predate Git, NetCDF conventions, and automated testing.

🔴 Config Fixed-Format

1

```
C START DATE          END DATE
C--------------  --------------
19780906060000  19780908060000
C
C PROP   U   SOURCE   U
C------- -   -------  -
5       M   10       M
C
C SPHER  SHALLOW  REFRA_D  REFRA_C
C------- -------  -------  -------
T       T        F        F
```

⚠️ Fixed-Format File for configuration data. Positional values, bare T/F booleans, ambiguous date strings. One misplaced space silently corrupts the run.

🔴 Cryptic Naming Style

2

```
SUBROUTINE CALC_V2B(N, NN, NNN)
  REAL    :: XTMP, XTMP2, XFINAL
  INTEGER :: IOPT, IPHYS, ISNL
  REAL    :: BETAMAX, FM, SIGMAA, SIGMAB

  ! ... 600 lines of mixed physics,
  !     dynamics and I/O logic ...
END SUBROUTINE
```

⚠️ CALC_V2B — version 2B of what? NN vs NNN — which is grid points, which is timesteps? XTMP and XTMP2 are used for completely different quantities. Context lives only in the original author's memory.

🔴 Dead Code Graveyard

3

```
! Original routine by J. Smith, 2003
!C SUBROUTINE OLD_PHYSICS_V1(ALPHA, FM)
!C ..
!C END SUBROUTINE
!
! Updated by K. Müller, 2009
!C SUBROUTINE OLD_PHYSICS_V2(...)
!C ... (150 lines)
!C END SUBROUTINE
! NOTE TO SELF: DO NOT DELETE
! MIGHT BE NEEDED AGAIN SOMEDAY
SUBROUTINE CURRENT_PHYSICS(...)
```

⚠️ Dead Code: "Just In Case" Graveyard. Layers of commented-out code from multiple authors across decades. Nobody knows if it still works. Nobody dares delete it. It clutters every search, every read, every code review.

🔴 I/O Custom Binary

4

```
C ABC PARAMETER OUTPUT IDENTIFIER
C---------+---------+
ABC
C
C XYZ OUTPUT IDENTIFIER
C---------+---------+
XYZ
C
! In source code (WAM_FILE_MODULE.f90):
OPEN(UNIT=20, FILE='ABC',    FORM='UNFORMATTED')
OPEN(UNIT=25, FILE='XYZ',    FORM='UNFORMATTED')
```

⚠️ Bare two-letter identifiers with no path or extension. Custom unformatted binary — unreadable without the exact Fortran reader. Output path lives hardcoded inside the source. Changing a filename requires recompilation.

✅ YAML

1

```
# model_config.yaml
run:
  start_date: "1978-09-06T06:00:00"
  end_date:   "1978-09-08T06:00:00"

timesteps:
  propagation: {value: 5,  unit: minutes}
  source_term:  {value: 10, unit: minutes}

grid:
  spherical:        true
  shallow_water:    true
  depth_refraction: false
```

Human-readable, order-independent, clean git diff, parseable by Python/shell/CI tools.

✅ Descriptive naming

2

```
SUBROUTINE compute_wave_energy_input( &
    n_grid_pts, n_freq_bins, n_directions)

  REAL    :: peak_frequency_hz
  REAL    :: phillips_parameter
  REAL    :: left_peak_width, right_peak_width
  INTEGER :: physics_scheme   ! 3=ST3, 4=ST4, 6=ST6

  CALL compute_wind_input(...)
  CALL compute_dissipation(...)
  CALL write_output(...)
END SUBROUTINE
```

Subroutine name describes its purpose. Variables are self-documenting. Single responsibility — one subroutine, one job.

✅ Using Git

3

```
# Delete it. Git has your back.
$ git log --all -- old_physics.f90
$ git show a3f92c1:src/old_physics.f90

# Tag the last working version before removal
$ git tag legacy/physics-v1 a3f92c1
$ git rm src/old_physics.f90
$ git commit -m "Remove deprecated physics routines
                 (recoverable via tag legacy/physics-v1)"eak
wave period Tp")
CALL nc_write(ncid, ...)
```

Version control is your safety net. History is preserved forever — there is no reason to leave dead code in the source.

✅ NETCDF/CF Conform

4

```
! NetCDF output with CF conventions
CALL nc_create("output/wave_fields.nc", ncid)
CALL nc_def_var(ncid, "significant_wave_height", &
                units="m",                       &
                long_name="Significant wave height Hs", &
                coordinates="lon lat time")
CALL nc_def_var(ncid, "peak_period",             &
                units="s",                       &
                long_name="Peak wave period Tp")
CALL nc_write(ncid, ...)
```

CF-compliant NetCDF is self-describing, readable by Python/R/CDO/NCO, portable across platforms, and interoperable with the entire ESM toolchain — no custom reader required.

The natESM Approach brings Research Software Engineering expertise into the heart of German ESM development. By embedding RSEs directly into modeling teams, the project tackles technical debt head-on — from untangling legacy structures to setting up CI pipelines and establishing sustainable coding conventions. Science continuity is always preserved; modernization happens one careful step at a time.