Lessons Learned from the OISST Software Rejuvenation, Implications, and Plans for Running Codes at NCDC

Drew Saunders
Operations Branch Chief

Outline

- Operations
  - Definition and Goals
- Research to Operations (R2O)
- OISST R2O

Operational?

NOAA's definition of *Operational*

Sustained, systematic, reliable, and robust mission activities with an institutional commitment to deliver appropriate cost-effective products and services

Sustained
Systematic
Reliable
Robust

mission activities with

Institutional commitment to deliver:

Cost effective

Appropriate

Products

Services

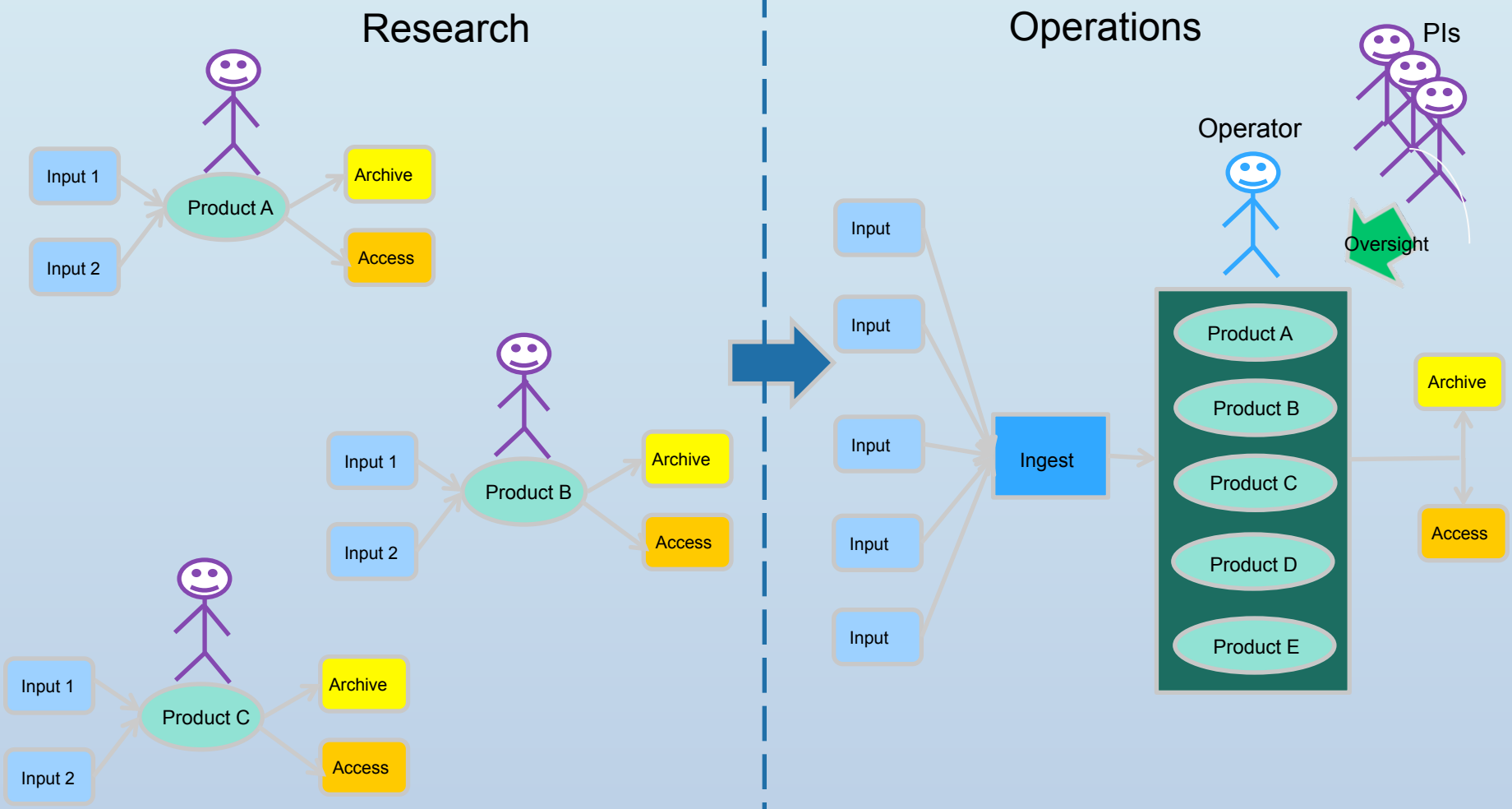
from National Oceanic and Atmospheric Administration (NOAA), 2009. Business Operations Manual, September 2009, version 5.2. National Oceanic and Atmospheric Administration.

http://www.ppi.noaa.gov//PPI_Capabilities/Documents/BOM.pdf

Operational Goals

- Reliable, Robust, Systematic, Sustainable
  - Environment (ITB, OB)
    - Controlled, automated, supported, utilizes a common infrastructure
    - Monitored: Processing and Products
  - Software (OB, PB)
    - Tested, documented, and maintainable
    - Configuration Management
  - Products (OB, PB, DAAB)
    - Scheduled production
    - Secure inputs
      - All climate duality products' inputs are archived
      - Service Level Agreements with input providers
    - Documented including Standard Operating Procedures
    - Defined level of service is affordable/reasonable: (not 24/7)
      - Climate quality (reliable) vs near real-time (best effort)
    - Reliable access
    - Quality contr

Research to Operations (R2O)



Transition from an environment with individual control to a shared automated environment with defined control …

OISST Technical Assessment

- Code base:
  - Approximately 11K LOC of F77, F90, Bash
  - 5 software components produce intermediate values
  - Multiple scripts with minor differences
- Evaluation criteria
  - Based on software industry best practices:
    - Code inspection: appearance, understandability, coding standards
    - Static analysis: unused code/variables, uninitialized variables, etc.
    - Metrics: lines of code, complexity
    - Performance
    - Importance of product and future plans
- OISST Recommendation: refactor the code
  - Performance not a major issue (~2 years/day)
  - Faster schedule and less risk than redesigning, rewriting, and testing entire code
  - Important product with PI retiring, future enhancements planned

OISST Results

- This slide is still being worked
- Compiler options
- Understand tool results
- Code cleanup
  - Lines of code

OISST Readability

Unrefactored

```
if (n.lt.2) go to 750 !(25 lines below)
    call decomp (x1,y1,s1,n,cond,ccos(j))
    if (cond.gt.cnmx) then
      mmax = ltmp - nsub(ltmp)
      iter = iter + 1
if (mmax.lt.0) then
print *,'RED-STOP! - OI-SSTMAP: mmax lt 0'
ierr = 1
return
endif
      go to 180 !(73 lines above)
    endif
    numdec = numdec + 1
    if (iter.gt.0) then
      if (iter.gt.10) iter=10
      niter(iter) = niter(iter) + 1
    endif
```

Refactored

Removed Goto

Added Comments

```
! If reasonable # of data points, decompose matrix

IF (num_data_points >= MIN_DECOMP_POINTS) THEN
  CALL decompose_coef_a(grid_box_stddev, lat_cos(this_global_lat), condition_code)

! If decomposition fails, reduce # obs.  Keep doing until < 0 obs
```

Improved Blocking

```
IF (condition_code > DECOMP_COND_THRESHOLD) THEN
  local_max_obs_used = num_points_to_check - obs_reduction_value(num_points_to…
  num_iters = num_iters + 1
  error_test = (local_max_obs_used < 0)
  error_msg = 'OI-CALC_SST_ANALYSIS_MAP: LOCAL_MAX_OBS_USED < 0'
  CALL oi_error_check(error_test, error_msg)
ELSE
```

Improved Variable Names

Modularized Repeated Code

```
  ! If decomposition succeeds, calculate X

  num_success_decomp_calls = num_success_decomp_calls + 1
  IF (num_iters > 0) THEN
    IF (num_iters > MAX_ITERS_BIN) THEN
      num_iters = MAX_ITERS_BIN
    ENDIF
    num_rgn_iters(num_iters) = num_rgn_iters(num_iters) + 1
  ENDIF
```

OISST Lessons Learned

- PI involvement critical
- Code metrics indicate 'better' quality code.
  - Lines-of-code (30% reduction for code)
  - Reduced cyclomatic complexity measure by 56%
    - Cyclomatic complexity indicates the number of unique paths through a function
      - Impacts testability and understandability
  - Cannot rely on metrics alone, need analysis to verify metrics
    - i.e. error checking can increase code complexity measure
- Testing
  - Unit testing (verifies functionality and behavior of a specific section of code)
    - Intermediate values allowed continuous testing during refactor phase.
      - Tools were written to support testing.
  - Integration testing (tests interfaces between components)
    - 30 day parallel test with original source code
  - Systems Test (tests requirements and functionality)
    - Need to perform System Acceptance Test before moving to operations
- Every research code/CDR is different
  - Assess to determine best value and most cost effective transition plan

OISST Lessons Learned (2)

- Non-operational benefits
  - Merged many copies of code into single configurable version controlled baseline
    - PI and developers can work in copies of same baseline
  - Automated builds that compile with gFortran or Lahey
  - Test procedures and data sets
  - Extracted common routines, smaller more understandable code