CRTM Status

Paul van Delst

Joint Center for Satellite Data Assimilation

Outline

- What is currently available
- CRTM Framework
  - Focus will be on why we chose this approach and how useful it has been so far.
- CRTM Requirements
  - Coding Guidelines
  - Code Acceptance Guidelines
- CRTM Repository

Current CRTM Status

- **`AtmAbsorption`**: Atmospheric absorption model
  - CompactOPTRAN
- **`AtmScatter`**: Obtains optical parameters from LUTs for both cloud and aerosol scattering and absorption.
  - `CloudScatter`: Water, ice, rain, snow, graupel and hail.
  - `AerosolScatter`: Dust, sea salt, dry/wet organic/black carbon, sulphate.
- **`SfcOptics`**: IR and MW models for four different surface types; land, water, snow, and ice
  - IR water: LUT based on Wu-Smith model.
  - IR land, snow, ice: LUT based on NPOESS database.
  - MW water: FASTEM-1.
  - MW land: Physical model (Weng and Yan).
  - MW snow, ice: Empirical models (Weng and Yan).
- **`RTSolution`**: Advanced doubling-adding (ADA) algorithm.

CRTM Framework

What is the CRTM Framework?

- At the simplest level, it's a collection of structure and interface definitions.
- There are User and Developer interfaces, as well as Shared Data interfaces and I/O.

Why do this?

- The radiative transfer problem is split into various components. Each component has its own structure definitions and application modules to facilitate independent development.
- Minimise or eliminate potential software conflicts and redundancies.
- Components developed by different groups can more easily be integrated into the CRTM for faster implementation of new algorithms.

CRTM Framework

Reality has exposed naïveté of this approach.

- Many research groups (ours included) do not have programming expertise and/or will to adhere to voluntary guidelines.

Lessons Learned. JCSDA Core CRTM Team needs to:

- Provide better documentation
  - Coding guidelines (done).
  - Code acceptance policy and procedures (draft).
  - Developer interface specification (still needs to be done).
- Provide access to the CRTM code base for developers.
  - Current CRTM repository is effectively inaccessible to members of even the JCSDA Core CRTM Team!
- Provide better direction to developers.
  - Fruitful communication between groups has not really occurred.
  - Hold regular code review meetings with developers. More specific direction is needed to let developers know if what they are doing will eventually be useful.

CRTM Requirements

CRTM Fortran95 Coding Guidelines

- Document is available.
- Tried to strike a balance between instruction and prescription.

CRTM Code Review and Acceptance Guidelines

- Draft is available. No input so far from management.
- Based on WRF/NMM policies and procedures.
- Main points:
  - Code review process
  - Code selection criteria: Speed, memory, accuracy. These are difficult to assign specific numbers as they can play off one another.
  - Repository organisation.
  - Repository access.
  - Code release policy.
  - Testing as part of acceptance (more on this later).

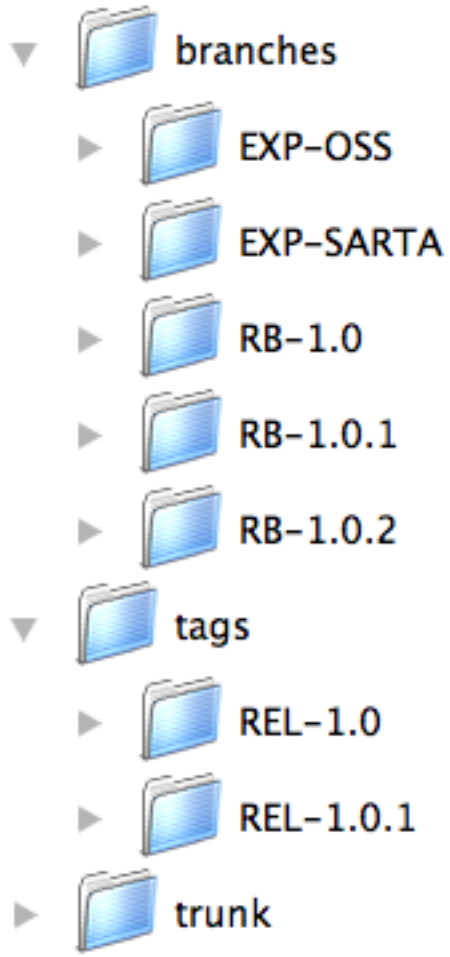
CRTM Testing

- This is one aspect of the CRTM that is woefully inadequate.
- We need to establish test cases that can be run to ensure code changes do not break existing code.
- Multiple approaches
  - Smoke testing: Catch “showstopper” defects in a CRTM component.
    - Fast.
    - Self-scoring.
    - Automatic as part of build.
  - Unit testing: Tests fine-grain elements of a CRTM component.
    - Self-scoring.
    - Automatic, or very simple to run.
    - Isolated. No interaction with other tests.
    - Self-contained. External changes do not affect results.
  - Regression testing:Test for unexpected consequences of integrating various CRTM components.

EMC CRTM Repository

- The EMC CRTM project subversion repository is located at
  `https://svn.ncep.noaa.gov/emc/crtm`
- Various other projects are maintained in the same repository.
- Write access to projects are determined by the project leader.
  - Requires a server account and password (via EMC helpdesk)
- Read access only requires you have access to the network on which the server resides.
  - Not a problem for people inside the NCEP firewalls.
  - **Big problem for those outside them.**
- The CRTM project contains three directories.
  - trunk: for mainline development
  - branches: for branch development off the mainline
  - tags: for labeling “frozen” snapshots of your system.

Current CRTM Repository Structure



- Three main directories
  - trunk
  - branches
  - tags
- trunk
  - The main line of development
  - Subdirectories here are based on typical convention: src, fix, scripts, external, and test.
- branches
  - This is where non-trivial development is done.
  - Experimental development branches: **EXP-***desc*.
  - Code release branches: **RB-***rel*.
- tags
  - This is where snapshots and release go.
  - Snapshots: **REL-rev***num*.*YYYY*-*MM*-*DD*.
  - Code releases: **REL-***rel*.
  - No development (otherwise it would be a branch.)

Future CRTM Repository

- Offsite server containing the CRTM repository.
- Hook repository up to an SCM[1] tool like Trac[2]. This is what the UKMO uses for its SCM so it's a proven tool for our application.
- A wiki for for online documentation.
- A user forum.
- ftp server
  - for downloading tarballs of releases (not all users will want to checkout the code from the repository.)
  - area for uploads
- "Hot-backup" for the repository
  - via post-commit hook scripts to a backup machine.
  - If primary server dies, we can simply switch to backup.
  - Restoration of web- and ftp-sites might take longer.
- Backup schedule. **Important!**
  - Incremental backups every night.
  - Full backups every week.
  - Backups kept for at least 1 month.

[1] SCM: Software Configuration Management

[2] See **`http://trac.edgewall.org`**