Extending SMRT

Towards a community model

Introduction

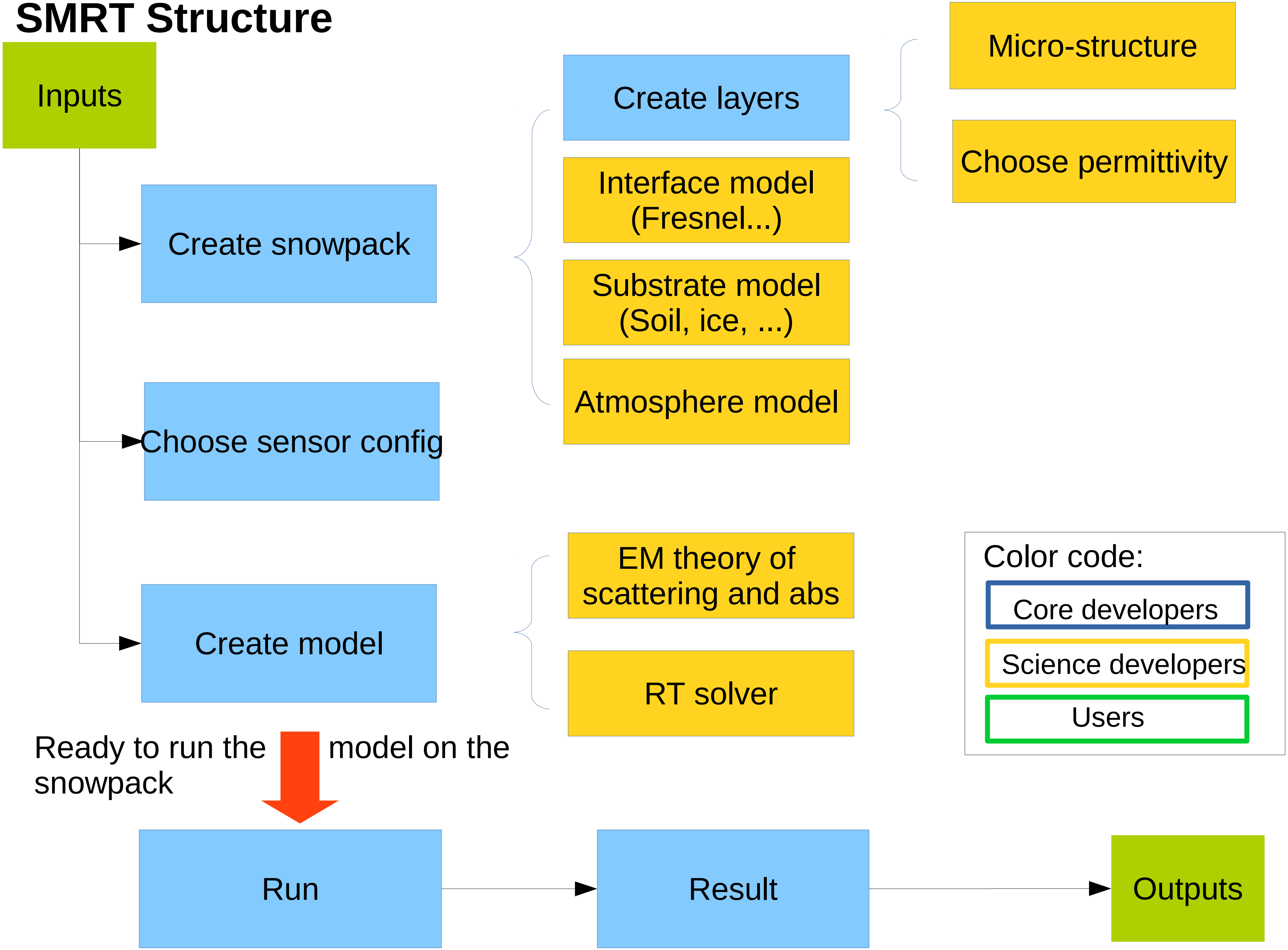
1 – Extending SMRT

Point of view: I need a new permittivity formulation or EM theory or RT solver or microstructure, ...

2 – Sharing developments

Point of view: I want to contribute to SMRT with my scientific dev

SMRT Structure



Inputs

Create snowpack

Create layers

Micro-structure

Choose permittivity

Interface model (Fresnel...)

Substrate model (Soil, ice, ...)

Atmosphere model

Choose sensor config

Create model

EM theory of scattering and abs

RT solver

Ready to run the model on the snowpack

Run

Result

Outputs

Color code:

Core developers

Science developers

Users

Science development

SMRT Structure ↔ directory structure

smrt/atmosphere	code to compute the Tbdown, trans et Tbup
smrt/core	forbidden ! The main machinery but no science
smrt/emmodel	electromagnetic code IBA, DMRT, Rayleigh…
smrt/__init__.py	forbidden ! were import start when « import smrt ».
smrt/inputs	user-oriented function to create snowpack, …
smrt/interface	code to compute R, T for inter-layer interfaces
smrt/microstructure_model	code with microstructure representation
smrt/permittivity	code with materials permittivity
smrt/rtsolver	code with RT Solvers
smrt/substrate	code to compute R, T for substrate
smrt/test	code to test smrt numerical results (using « nosetest »)
smrt/utils	various utilities related to smrt : wrappers to other Models, plotting functions, …

Science development

Recommended way to extend SMRT: Create a new file in the relevant directory, that's it !

E.g. to add a microstructure :

1. Copy iba.py my_super_scatt_theory.py
2. Edit my_super_scatt_theory.py
3. Ready to use and intercompare: m = make_model(« my_super_scatt_theory », « dort »)

No need to compile anything or create a configuration file. New files are automatically discovered.

Rmq :

Create new files, **do not modify existing files**.

→ Keep the compatibility : « git pull » works to get updates. Easy to transfer to someone, just email the new file and in which directory to put it. Your colleagues is ready to go !

Rmq :

To test variants : copy iba.py improved_iba.py, make the change, and

m1=make_model(« iba », « dort »)

m2=make_model(« improved_iba », « dort »)

I've optimized or developed most part of SMRT like this, step by step keep a « reference » slow code.

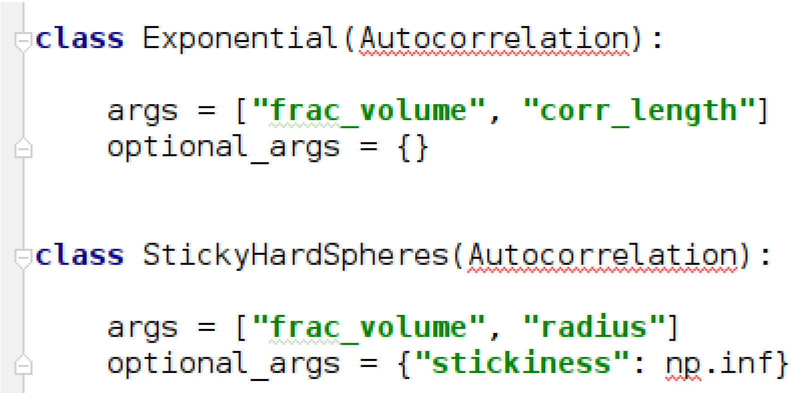
Science development

E.g.

Create a new file in the relevant directory, that's it !

E.g. to add a microstructure :

1. Copy exponential.py mysupermicrostructure.py
2. Edit mysupermicrostructure.py – **add your specific arguments**
3. Ready to use: sp = snowpack(thickness, « mysupermicrostructure », —---------------)



```
class Exponential(Autocorrelation):

    args = ["frac_volume", "corr_length"]
    optional_args = {}


class StickyHardSpheres(Autocorrelation):

    args = ["frac_volume", "radius"]
    optional_args = {"stickiness": np.inf}
```

Towards a community model

Sharing your scientific developments in SMRT is more than welcome, especially for published works.

Objective: Extend as much as possible
while maintaining quality

<u>Ideal requirements:</u>
- exactness and broad interest of the code
- clean code following guidelines and documentation
- sustainibility and a vision/roadmap (backward compatibility, no overlap, ...)

<u>Several levels of maturity:</u>

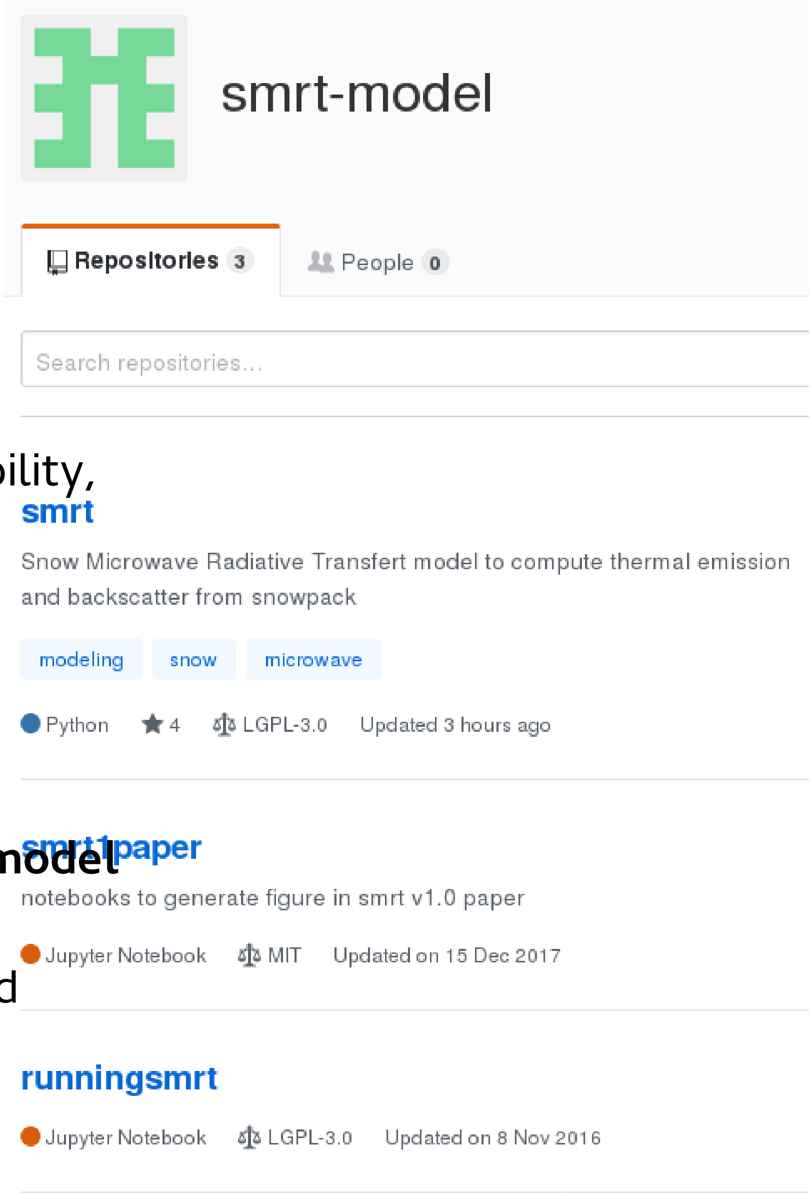
1- In a public repository on your own (github, ...)

2- In a "user-contrib" repository on **github smrt-model**

3- Integration in SMRT codebase itself on **github smrt-model**

The mechanisms of auto-discovery need/will be extended

Most likely:
smrt.path.append(« ~/smrt_usercontrib »)



smrt-model

Repositories 3 | People 0

Search repositories...

smrt

Snow Microwave Radiative Transfert model to compute thermal emission and backscatter from snowpack

modeling | snow | microwave

● Python ★ 4 LGPL-3.0 Updated 3 hours ago

smrt1paper

notebooks to generate figure in smrt v1.0 paper

● Jupyter Notebook MIT Updated on 15 Dec 2017

runningsmrt

● Jupyter Notebook LGPL-3.0 Updated on 8 Nov 2016

Towards a community model

SMRT coding rules:

- explicit names for file, class and variable (lowercase word separated by _). Relax for very local variables. Names must be clear and non ambiguous. Avoid abbreviations. Short is better than long, but explicit is always better than implicit.

- make the functions and classes as general as possible + use option arguments with default values for the most widely "expected behavior".

- use S.I. unit without multiplictor or divisor: m, kg, s, Hz

- code formatted using PEP8 (with some rules relaxed)

- documentation directly in python code → autogenerated to readthedoc.io

- write unit test (files starting with test_) for every piece of code.

Towards a community model

Roadmap or how you can effectively help:

- read, comment and edit the online documentation. Adding refs, more explanations
- write tutorials or organize training
- add pre-defined sensors (easy, can be done today !)
- add permittivity formulations for ice and other materials (e.g. Turi's formulation)
- add soil models for passive (e.q. QNH model, see DMRT-ML) and active (e.g. AIEM python?)
- add HUT atmosphere or other simple model
- code review, writing unit test.

A bit more involved:

- add new media (e.g. sea-ice layer, forest layer, multi-layer atmosphere):
    1) need slight core changes (I'll do it soon!)
    2) sp = make_snowpack → si = make_seaice   medium = sp + si
    f = make_forest → medium = sp + f

- add RT solvers:
    1) 6-flux (nearly finish)
    2) DORT with coherent layers (C. Matzler appraoch)
    3) solver for altimetery (first order)
    4) solver for birefringent media (needs slight core changes)
- make an online version like https://snowtartes.pythonanywhere.com