SWIMMR

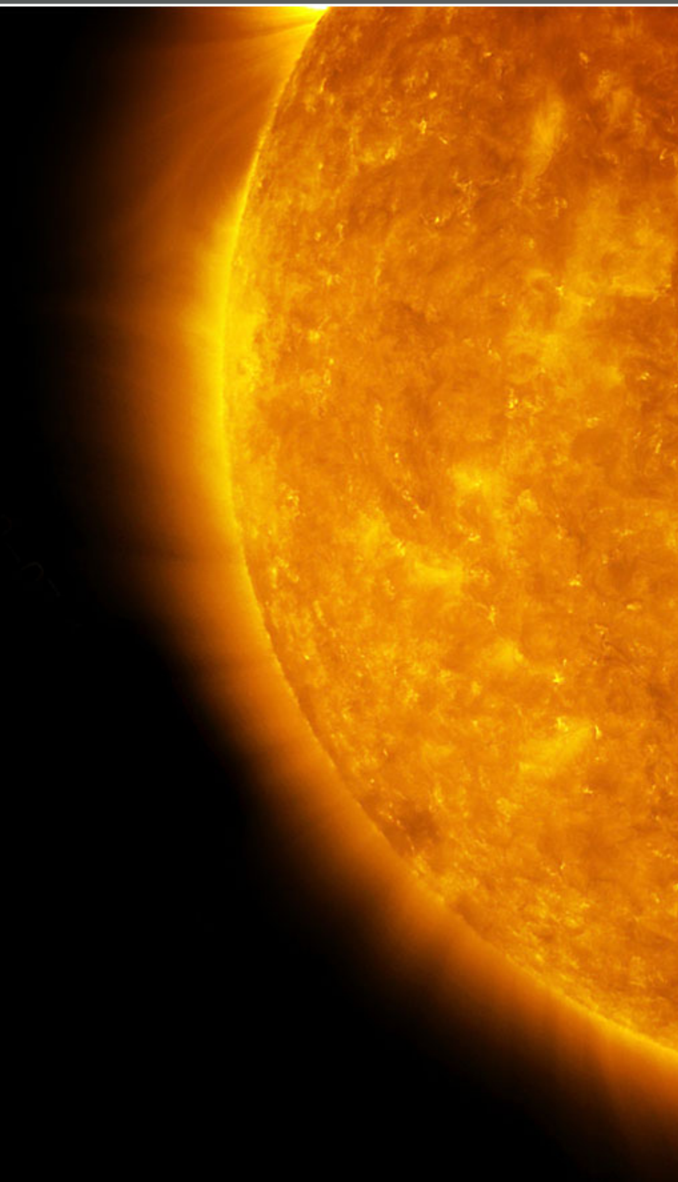
Requirements for Operations

David Jackson

Met Office - Manager of Space Weather Research

19th December 2019

Thanks to Mike Marsh and Edmund Henley for a lot of the slides



@MetOfficeSpace

Overview

- Solterra project – summary of (scientific+) + user + system requirements for Met Office S2E modelling system
  - Example using 2 whole atmosphere models
- User and System (IT) Requirements modified for SWIMMR
- Platforms
  - AWS
  - HPC (MonSoon)

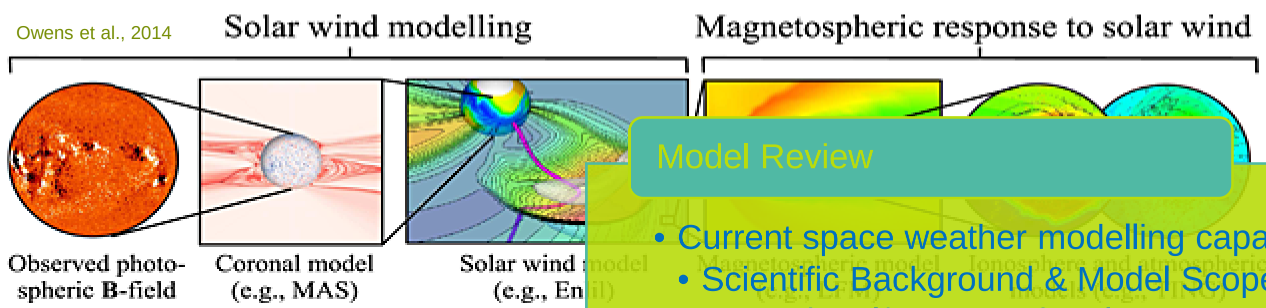
Sol-Terra Overview of Approach

Review Available Space Weather Models

- Categorise by region in domain (eg ionosphere)
- Scientific Review
- Operational Review – user and system requirements

Scientific Domain Overview

Owens et al., 2014



Model Review

- Current space weather modelling capabilities
  - Scientific Background & Model Scope
  - Operational/Computational Aspects
  - Inputs & Coupling
  - Operational Sustainability

Operational Forecast Potential

- **Operational Constraints**
  - **Time Constraints** – Computational/Data Latencies
  - **Source Code Quality Assurance Factors**
    - Documentation Standards
    - Version Control (e.g. Git, Subversion)
    - Error Handling
    - Languages, Dependencies
- **Verification & Ensembles**

Met Office User & System Requirements for Solterra

User Requirements

- **Timeliness**: Data assimilated and forecasts produced in
- **Data Model**: data driven climatology.
- **User Documentation**: High standard – describing model overview, input/output and limitations on validity.
- **Evaluation**: Model statistical verification skill scores defined to inform forecaster interpretation.
- **Ensembles**: possible
- **Autonomy**: potential to run automatically

User

Space Weather Operations Centre (MOSWOC)

Space Weather Operational Meteorologist

System

MOSWOC Infrastructure

System Requirements

- **Robustness**: models should run successively for a range of space weather conditions
- **Forecast Cycle**: The model should run fast enough to be used within a forecasting cycle
- **Quality Assurance**: High standard of code structure, documentation, error handling and version control allowing systematic model management
- **Environment**: Developed using appropriate operating system (for Met Office this is Linux).
- **Language**: Source code available and model written in appropriate language (for Met Office: Fortran … or Java).
- **Licence**: …se, IPR and terms of use for model … data should be appropriate and obtainable.
- **Efficiency**: If needed, code should be parallelised to … operation
- **Resilience**: Fall back option of using a simpler configuration, other initialisation, repeat forecast, or alternative input data source to maintain continuous forecasting capability (e.g. solar or geomagnetic drivers as input) in case of technical issues
- **Dependencies**: Models should not have dependencies on non-standard libraries not under the operational centre control
- **Coupling**: Model should be suitable for coupling to other appropriate models

Example for Whole Atmosphere Models

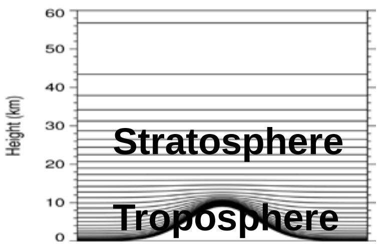
Whole Atmosphere Model (WAM)

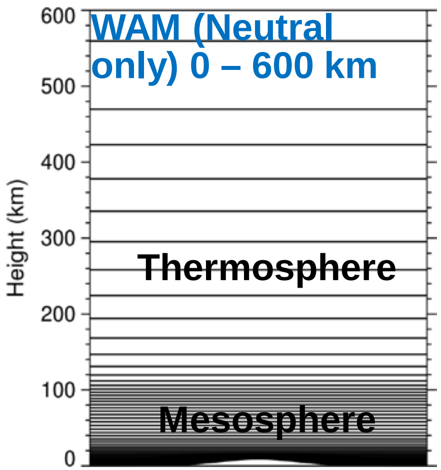
- $p_{top} = 1.5 \times 10^{-7}$ Pa (O exobase)
- T62L150 (~ 2°×2°, ~ 0 – 600 km)
- Free runs or A/F cycle (NWP-DA + MA Data 60-100km)
- Composition dependent R & C_p
- Height dependent $g(z)$

Physics

- Horizontal & vertical mixing (no “sponge”)
- Radiative heating: EUV, UV, & non-LTE IR
- Major neutrals (O, O_2, N_2)

GFS 0 – 60 km





Canadian Ionosphere and Atmosphere Model (C-IAM)

- C-IAM describes the neutral atmosphere, ionosphere and inner magnetosphere up to ~15 Earth radii, in a self consistent manner. Based on CMAM neutral atmosphere model
  - T47 L95 (ground to 2×10^{-7} hPa (200–350 km depending on solar cycle, location, season and local time).
- Hydrostatic. Full dynamics, radiation (including non-LTE), neutral chemistry, ion chemistry Self-consistent calculation of E field
- C-IAM currently under development.

Met Office

User/System Requirements assessment

	User Requirements						System Requirements											
Model Nam	**UR-1.** Timeliness	**UR-2.** Data Model	**R-3.** User Documentation	**UR-4.** Evaluation	**UR-5.** Ensembles	**UR-6.** Autonomy	**SR-1.** Robustness	**SR-2.** Forecast Cycle	**R-3.** Quality Assurance	**SR-4.** Environment	**SR-5.** Language	**SR-6.** Licensing	**SR-7.** Efficiency	**SR-8.** Resilience	**SR-9.** Dependencies	**SR-10.** Coupling	**SR-11.** Data Guidlines?	**Operational Analysis**
WAM	Met	Met	Met	Partial	Met	Met	Met	Met	Met	Met	Met	Met	Met	Met	Partial	Partial		Suitable
CIAM	Met	Met	Unmet	Unmet	Met	Unmet	Met	Met	Unmet	Met	Met	Unmet	Unmet	Met	Met	Partial		Unsuitable

Accept =
Further Development =
Reject =

•WAM satisfies many of the requirements . NOAA GFS heritage means high standards of code documentation and robustness. Evaluation, software dependencies and coupling are areas needing further development.

•C-IAM satisfies a number of the requirements. It is a relatively new model, and also has no external users, so some of the requirements are less well developed. User documentation, evaluation, efficiency and coupling need further development. Critical requirements (autonomy, QA, licencing) not met.

Met Office

Met Office User & IT Requirements for SWIMMR

User Requirements

- **AssimilationTimeliness**: where data assimilation is required, data assimilated in(NRT)
- **Forecast Timeliness**: forecasts with an appropriate timeliness
- **Forecast Validity:** Forecasts shall be produced with a validity appropriate to the application.
- **Model Robustness:** Models shall run successfully and generate realistic outputs for a range of space weather conditions. Consideration of extreme weather conditions should be incorporated wherever possible.
- **Model Type**: Models should be data driven rather than climatology, unless infeasible or inappropriate
- **User Documentation**: The models shall be accompanied by a description and guide to model use and limitations, supplemented with references to associated published papers
- **Model Skill**: Model statistical verification skill scores defined and recorded to inform forecaster interpretation. and benchmarking
- ***Ensembles**: Ensemble operation possible (unless deterministic models are shown to perform better)*.
- **Autonomy**: Models should have the potential to run automatically, without human intervention

System Requirements

- **Robustness**: models should run successively for a range of space weather conditions and handle errors appropriately and informatively, allowing operational service and IT support teams to understand and resolve problems.
- **Forecast Cycle**: The model should run fast enough to be used within a forecasting cycle excluding data latencies (varies by domain and conditions).
- **Quality Assurance**: The proposal shall be able to demonstrate that the model will be developed and delivered to a good standard, with version control, error handling, code review and acceptance testing.
- **IT Robustness and Error logging**: Models shall be written to an acceptable standard and shall handle errors appropriately and informatively, allowing Operational Service and IT Support teams to both understand and resolve problems
- **Environment**: Models shall be developed to run under the Linux operating system at the Met Office, for both on-premises and Amazon Web Service (AWS) operation
- **Language**: Source code shall be delivered and model written Fortran, Python or Java. C ~~and IDL~~ may also be acceptable for critical models, Appropriate versions of the above languages shall be discussed with the Met Office before submission of the proposal.
- **Data Licensing**: In addition to a royalty free, non-exclusive licence for any model developed, licenses and terms of use for any input data shall be obtainable and described.
- ***Efficiency**: For computationally expensive models which are chosen to run on the Met Office High Performance Computer (HPC) rather than AWS, code should be parallelised to ensure HPC operation (supported OpenMP and MPI protocols).*
- **Resilience**: In case of data or model failure due to technical issues, fall back option of using a simpler configuration, other initialisation, repeat forecast, or alternative input data source to maintain continuous forecasting capability
- **Software Libraries**: Models should not have dependencies on non-standard libraries not currently supported by the Met Office. Applicants are advised to contact the Met Office to discuss this.
- ***Data Requirements**: All input and output data requirements should be clearly stated to help ensure the highest level of compatibility with other SWIMMR projects.*

Don't worry - in more detail overleaf..

Met Office

User Requirements for SWIMMR

User Requirements

- **AssimilationTimeliness**: where data assimilation is required, data assimilated in(NRT)
- **Forecast Timeliness**: forecasts with an appropriate timeliness
- **Forecast Validity:** Forecasts shall be produced with a validity appropriate to the application.
- **Model Robustness:** Models shall run successfully and generate realistic outputs for a range of space weather conditions. Consideration of extreme weather conditions should be incorporated wherever possible.
- **Model Type**: Models should be data driven rather than climatology, unless infeasible or inappropriate
- **User Documentation**:The models shall be accompanied by a description and guide to model use and limitations, supplemented with references to associated published papers
- **Model Skill**: Model statistical verification skill scores defined and recorded to inform forecaster interpretation.
- ***Ensembles**: Ensemble operation possible (unless deterministic models are shown to perform better).*
- **Autonomy**: Models should have the potential to run automatically, without human intervention,

Obvious, for a focused NRT operational system

For most projects, prototype runs (on AWS) for ~12 months. Still solar min - may also need extreme event tests

Should be useful for forecasters – but not BCS standard.

Verification becoming standard for MO; should be for you too. If you don't beat the existing system your system will be rejected

Desirable – no explicit need for ensembles here

Obvious, for a NRT operational system

IT Requirements for SWIMMR (I)

Need robustness and traceability hence vn control and acceptance tests vital

Do you want a call from us at 3am on a Sunday?

See next talk from Alex for AWS

Matches skills of 24/7 IT Support

System Requirements

- **Quality Assurance**: The proposal shall be able to demonstrate that the model will be developed and delivered to a good standard, with version control, error handling, code review and acceptance testing.
- **IT Robustness and Error logging**: Models shall be written to an acceptable standard and shall handle errors appropriately and informatively, allowing Operational Service and IT Support teams to both understand and resolve problems
- **Environment**: Models shall be developed to run under the Linux operating system at the Met Office, for both on-premises and Amazon Web Service (AWS) operation
- **Language**: Source code shall be delivered Fortran, Python or Java. C ~~and IDL~~ may also be acceptable for critical models, Appropriate versions of the above languages shall be discussed with the Met Office before submission of the proposal.
- **Data Licensing**: In addition to a royalty free, non-exclusive licence for any model developed, licenses and terms of use for any input data shall be obtainable and described.

Met Office

IT Requirements for SWIMMR (II)

Desirable_
Overriding req. for HPC models is that forecasts are completed in NRT.

MOSWOC still need to issue a forecast so void needs filled – should include this resilience in initial system design

Packages well-supported by community ~OK (e.g. Astropy), tiny ones not (e.g. MyCode.py, 1 dev did 3 commits 5 years ago). HPC: more restrictions. Tension with maintainability of "roll-your-own" acknowledged!

Desirable (essential for a coupled S2E system, but this not being built here)

Also: Python 3 please!

System Requirements

- ***Efficiency***: *For computationally expensive models, which are chosen to run on the Met Office High Performance Computer (HPC) rather than AWS, code should be parallelised to ensure HPC operation (supported OpenMP and MPI protocols).*
- **Resilience**: In case of data or model failure due to technical issues, fall back option of using a simpler configuration, other initialisation, repeat forecast, or alternative input data source to maintain continuous forecasting capability
- ~~**Dependencies**~~ **Software Libraries**: Models should not have dependencies on non-standard libraries not currently supported by the Met Office. Applicants are advised to contact the Met Office to discuss this.
- ***Data Requirements:*** *All input and output data requirements should be clearly stated to help ensure ... MMR*

Recommendations for effective R2O

Input data: use real-time sources, not science-grade sources

- Less OMNI, more SWPC (datagaps, bad data etc)

Version control (GitHub, git, svn, …)

- Public / private – ideal if MO can access repo
- Use meaningful commit messages
- Create branches for new features (a la GitFlow)
  - Don’t forget to merge back to trunk when happy!

Use a ticket system – one ticket per **small** change

- Gives extra context to commits – ideal if MO can access too
- Ideally get someone to review/QC ticket before merge

Have at least a basic test harness

- If retrofitting: add end-to-end (system) test(s) / integration tests
  - Same input: does model still give same output?
- Ideally add (faster, smaller) unit tests for any new code
  - Same input: does function still give same output?

Run tests on commits to the repo

- Ideally automatic each time – continuous integration
  - E.g. via Jenkins, Travis-CI, Circle-CI (often free if open source)

Use logs / stdout / stderr / return codes effectively

- Default to quiet: easier to see warnings/errors (don’t ignore these!)

Don’t let perfect be the enemy of good! A little goes a long way

- I’ve run tests manually on commits – better than none!
- We’ve not always adhered to above – have been bitten!
- We’re adding in as much as possible, being pragmatic

Met Office

Cautionary tales

```
Input data: use real-time sources; have tests for handling code!
# Branch in code given to us, with no associated test
# We did not spot this, only hit buggy code operationally when L1
# switched from DSCOVR to ACE and started getting NaN values in x…
if np.isnan(x) and (y != 0):  # Presumably *intended* to handle ACE
    # buggy code, fails if run – presumably never even tested by hand
```

Version control and tickets

- Repo access: received lots of code via email / similar. Convenient at time; awful long-term. E.g. find a bug: hard to find where introduced, relevant commits in parent repo, if fixed there…
- Commit messages: trying to debug code last changed 4 years ago, original devs left MO:
  - "`coming along nicely`" [4 new files, 2 changed files]
  - "" [This commit is too large to render. Showing the first 1000 files…]
  - Only saving grace – there were tickets, which gave context to what was being done!
- Branches and merging: branch shared with us (DVD by post), owned by postdoc. Never merged, difficulties picking up with code owners after postdoc left.
- Atomic tickets/commits: I often do solo work; used to bundle lots of changes in vague tickets Changing since working more with others / having to revisit my old work and untangle things!

Test harnesses

- Lack of end-to-end test covering a code translation from IDL to Python has made it very hard to pin down where a bug affecting extreme space weather values is, and where has come in
- Adding tests has made porting Python code from on-prem to AWS easier, more robust

Quiet logs, and warnings

- Prints makes lots of code unhelpfully verbose – harder to spot warnings
- Change compiler gfortran ↔ ifort leads to crashes after days; ≥10% differences. No –Wall?

Met Office

The big picture

Inversion

TxWx SpWx

NMHS support

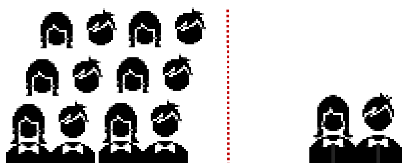
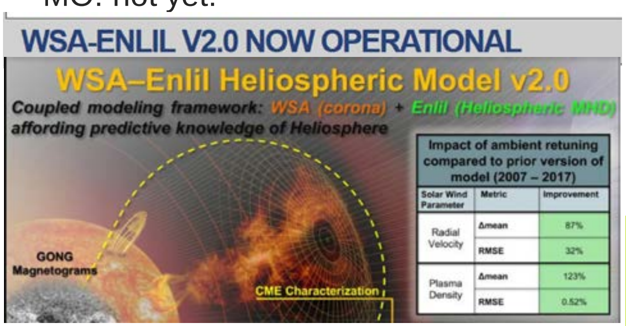


Table 2
Model Upgrade Histories

MacNeiss 2018

Model	Version	Release date (to CCMC)
WSA	1.6	September 2006
	2.2	April 2010
	3.6	March 2014
	3.7	December 2015
	4.0	September 2016
	4.2	February 2017
	4.3	September 2017
	4.4	February 2018
	4.5	June 2018

Operationalisation bottleneck
SWPC 2019: *“the first upgrade since the initial implementation in late 2011”*
MO: not yet!



Skill set

R2O/reproducibility toolbox is useful for many careers:
- Research
- Data scientist
- Quant
- Software engineer…

…

And it’s not hard to learn

Avoid!

Different dev/prod envs
Testing a mess
Help avoid this being us!

software carpentry
Teaching basic lab skills for research computing

Operational Architecture

Current
- Expensive models (WSA Enlil, SWMF) use HPC
- Other models largely use Virtual Linux Machines

Future
- Expensive models will likely still use HPC
- Other models being moved to AWS (or will be implemented there)

For SWIMMR
- HPC access via MonSoon – see next slide
- Shared Met Office / academic space on AWS
    - AWS database of operational and new observations to be developed
    - Models to be implemented on AWS – “researcher sandbox”
    - Prototype system running on AWS for several months before end of SWIMMR – and later R2O much easier since AWS setup the same
    - See Alex’s talk (next) for much more detail

MonSoon

- Part of Met Office Supercomputer open to academic community
- MonSOON and operational HPC setup the same – reduces R2O overheads
- Free for NERC funded projects
- Application made after funding awarded

https://www.metoffice.gov.uk/research/approach/collaboration/jwcrp/monsoon-hpc

Any Questions?