Norwegian
Meteorological
Institute

JSON schema validation of experiment configurations

Joint 30th Workshop All-Staff Meeting 2020

Roel Stappers
Met Norway
roels@met.no

02.04.2020

Background/Motivation

- In Harmonie-Arome we currently validate only a fraction of our configuration options, e.g.
  - TSTEP needs to be a divisor of 3600
  - NLON, NLAT needs to be of the form $2^a 3^b 5^c$

- For many variables the allowed values are only mentioned in comments
- We override values based on other configuration options.
- Waste of SBU's by invalid configurations.

- See also RWP: SY3 clean up of Harmonie scripting system
- In addition we have a growing number of configurations in our systems (EPS, reanalysis, climate, high-res, nowcasting)

We need a more structured way to handle and validate configuration settings in Harmonie-Arome.

Configuration in Harmonie

Very roughly speaking the Harmonie scripting system consists of 2 parts

1. Scripts called by ecflow tasks (in ecf/ and scr/ directory (CY43))
2. data files to handle configuration *,
    - config_exp.h
    - include.ass
    - Harmonie.pm
    - harmonie_namelists.pm
    - ECFlow Suite definitions (tdf files)
    - Configuration for submission (submit.LinuxPC submit.ecgb, etc.)
    - Configuration for HPC environments (config.ecgb etc.)
    - Configuration for VarBC
    - Information on ECMWF cycles (MARS parameters etc)
    - Configuration for blacklisting
    - Harmonie_domains.pm
    - Harmonie_configurations.pm
    - Harmonie_testbed.pl (test_defs part)

* These files not pure data files but also contain Perl code.

Norwegian
Meteorological
Institute

Current status.

- Work has started to use a common format for all configuration files.
- Language independent (TOML). Possible alternatives would be YAML or JSON.
- config_exp.toml is finished (for deterministic).
- Next step will be include.ass and harmonie.pm

- Configurations are validated using JSON Schema (next slide)
  - Implementation exist for a wide range of languages (C/C++, Go, Java, PHP, Javascript, Python,Perl, etc)
  - Supported by several editors/IDEs
  - Automatic creation of GUIs

Norwegian
Meteorological
Institute

config_exp.toml

```
# **** Model geometry ****
[Geometry]
DOMAIN='DKCOEXP'                      # See definitions in scr/Harmonie_domains.pm
TOPO_SOURCE='gmted2010'               # Input source for orography. Available are (gtopo30|gmted2010)
                                      # For usage of gmted2010 check the documentation first
GRID_TYPE='LINEAR'                    # Type of grid (LINEAR|QUADRATIC|CUBIC)
VLEV='65'                             # Vertical level definition.
                                      # HIRLAM_60, MF_60,HIRLAM_40, or
                                      # BOUNDARIES = same number of levs as on boundary file.
                                      # See the other choices from scr/Vertical_levels.pl

VERT_DISC='vfd'                       # Discretization in the vertical (vfd,vfe)
LGRADSP='yes'                         # Apply Wedi/Hortal vorticity dealiasing (yes|no)

# **** High level forecast options ****
[Physics]
DYNAMICS="nh"                         # Hydrostatic or non-hydrostatic dynamics (h|nh)
PHYSICS="arome"                       # Main model physics flag (arome|alaro)
MASS_FLUX_SCHEME='edmfm'              # Version of EDMF scheme (edkf|edmfm)
                                      # Only applicable if PHYSICS=arome
                                      # edkf is the AROME-MF version
                                      # edmfm is the KNMI implementation of Eddy Diffusivity Mass Flux scheme for Meso-
STATNW="yes"                          # Switch for new set up cloud sscheme (yes|no)
HARATU="yes"                          # Switch for HARATU turbulence scheme (yes|no)
ALARO_VERSION=0                       # Alaro version (1|0)
XRIMAX=0.0                            # Maximum allowed Richardson number in the surface layer (cy40h default was 0.0)
```

JSON Schema (example 1)

```
"CISBA": {
     "type": "string",
     "description": "ISBA scheme",
     "enum": [
       "3-L",
       "2-L",
       "DIF"
     ],
     "default": "3-L",
     "links" : [
       {
         "rel" : "ISBA documentation",
         "href" : "https://www.umr-cnrm.fr/isbadoc/model.html"
       }
     ]
   }
```

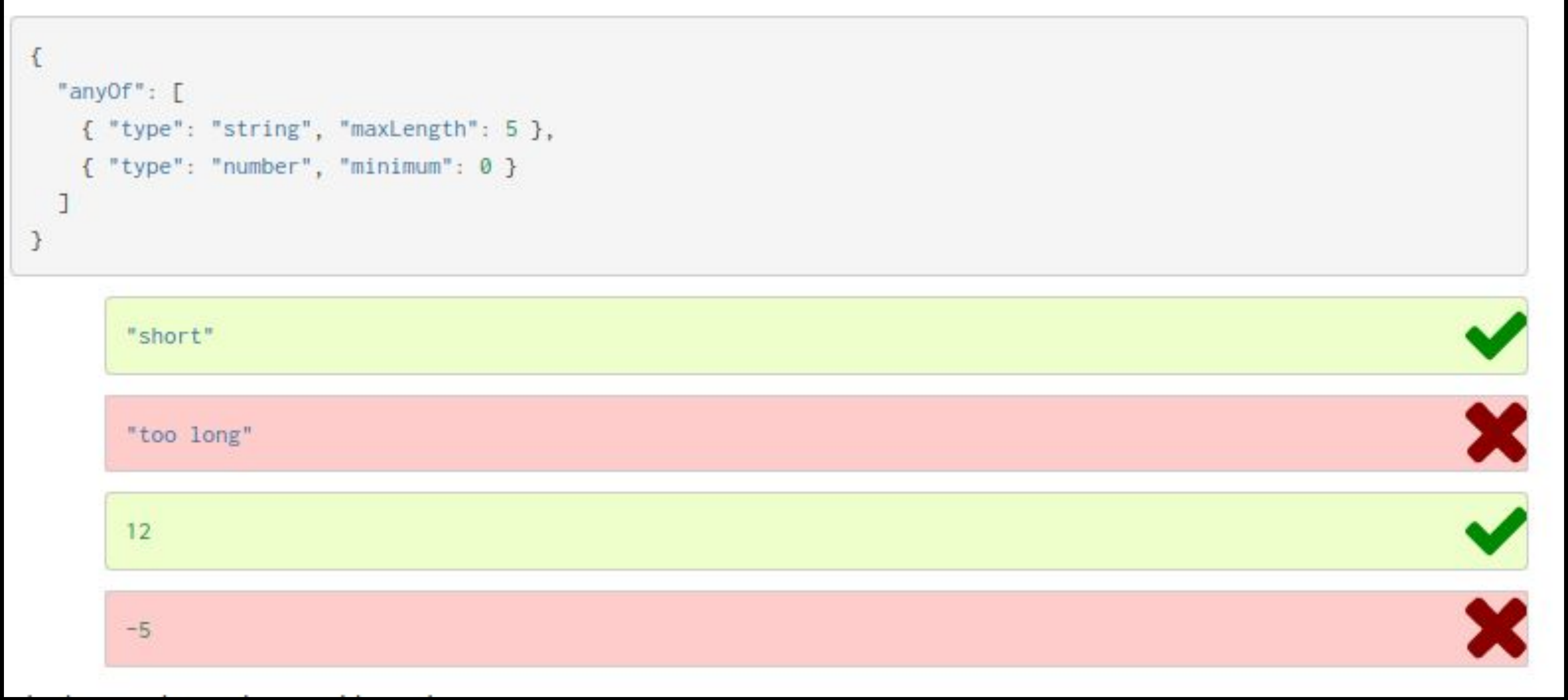
JSON Schema (example 2)

```
"CROUGH": {
     "type": "string",
     "description": "type of orographic roughness length",
     "enum": [
       "NONE",
       "Z01D",
       "BE04"
     ],
     "options" : {
       "enum_titles" : [
           "NONE | no orographic treatment",
           "Z01D | orographic roughness length does not depend on wind direction",
           "BE04 | Beljaars 2004 orographic drag"
       ]
     },
     "default": "NONE",
     "links": [
       {
        "rel": "CROUGH Surfex documentation",
        "href": "http://www.umr-cnrm.fr/surfex/spip.php?article126"
       }
     ]
   }
```

Norwegian
Meteorological
Institute

JSON Schema.

Combining Schema’s. anyOf



```
{
  "anyOf": [
    { "type": "string", "maxLength": 5 },
    { "type": "number", "minimum": 0 }
  ]
}
```

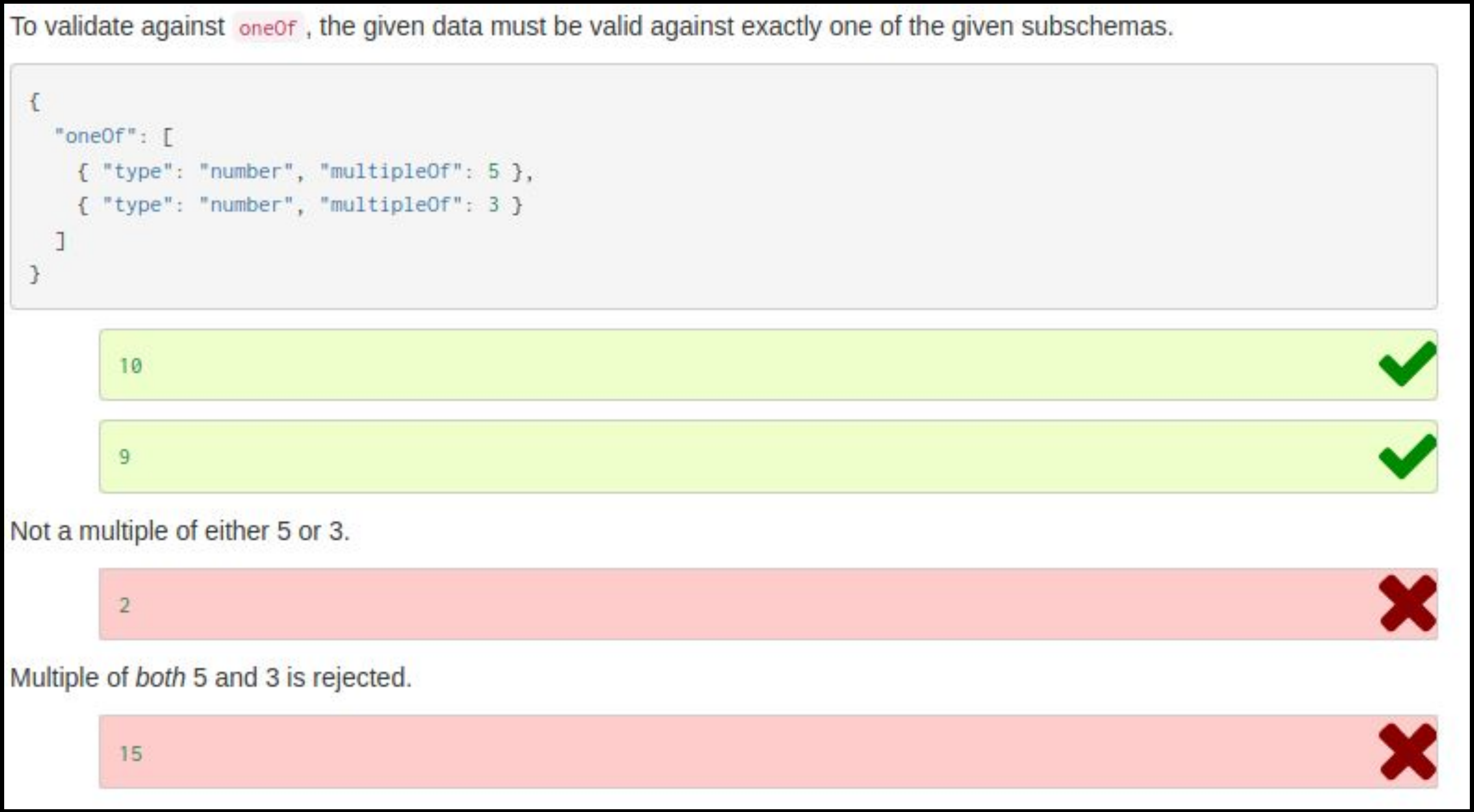
```
"short"
```
✓

```
"too long"
```
✗

```
12
```
✓

```
-5
```
✗

JSON Schema (Example)



To validate against `oneOf`, the given data must be valid against exactly one of the given subschemas.

```
{
  "oneOf": [
    { "type": "number", "multipleOf": 5 },
    { "type": "number", "multipleOf": 3 }
  ]
}
```

```
10
```
✓

```
9
```
✓

Not a multiple of either 5 or 3.

```
2
```
✗

Multiple of *both* 5 and 3 is rejected.

```
15
```
✗

JSON Schema

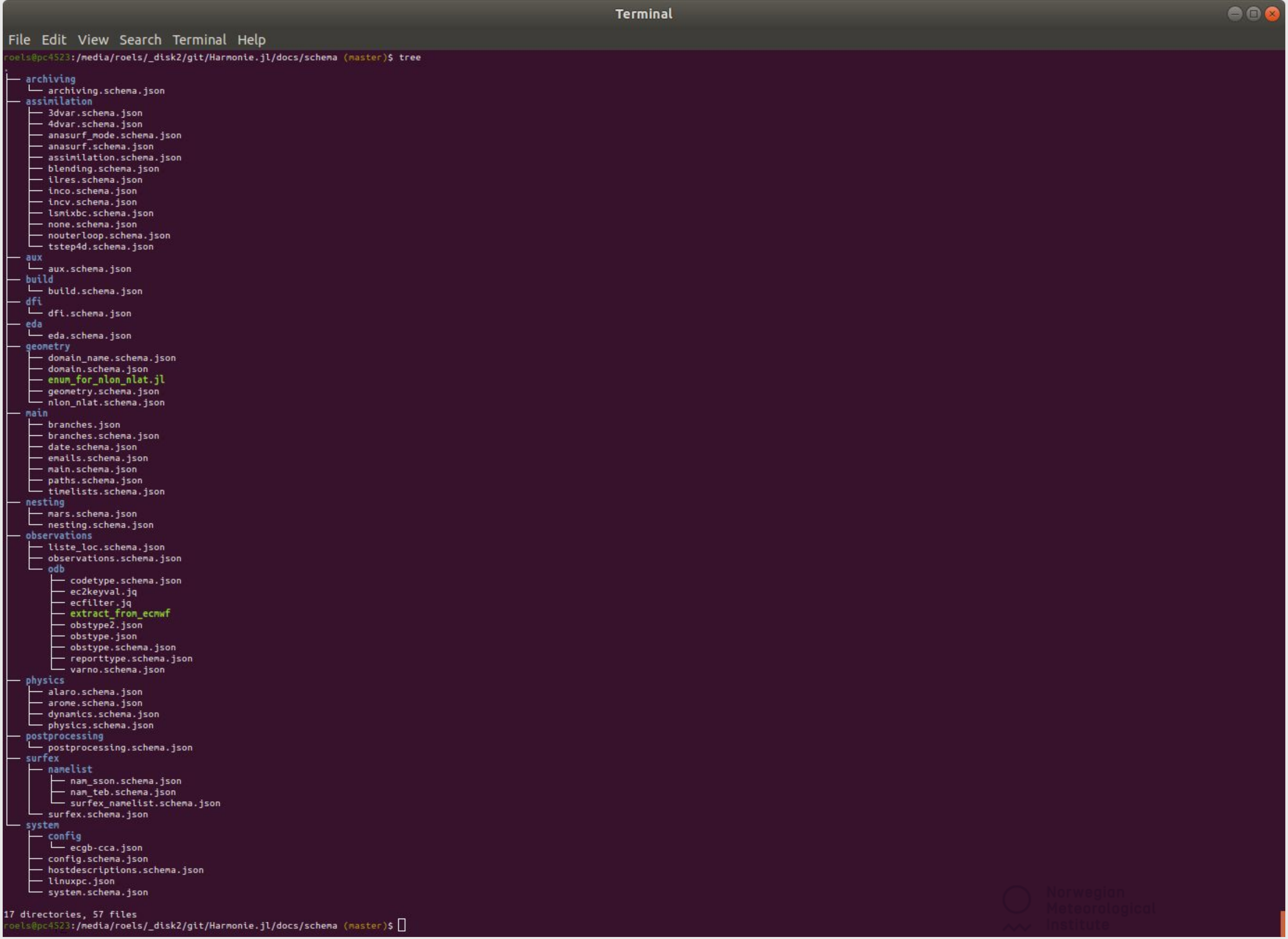
assimilation.schema.json

```
{
  "type": "object",
  "oneOf": [
    {
      "title": "3DVAR",
      "$ref" : "3dvar.schema.json"
    },
    {
      "title" : "4DVAR",
      "$ref" : "4dvar.schema.json"
    },
    {
      "title": "blending",
      "$ref" : "blending.schema.json"
    },
    {
      "title": "none",
      "$ref" : "none.schema.json"
    }
  ]
}
```

Norwegian
Meteorological
Institute

Validation using JSON Schema

See https://json-schema.org/ for full specification



```
roels@pc4523:/media/roels/_disk2/git/Harmonie.jl/docs/schema (master)$ tree
.
├── archiving
│   └── archiving.schema.json
├── assimilation
│   ├── 3dvar.schema.json
│   ├── 4dvar.schema.json
│   ├── anasurf_mode.schema.json
│   ├── anasurf.schema.json
│   ├── assimilation.schema.json
│   ├── blending.schema.json
│   ├── ilres.schema.json
│   ├── inco.schema.json
│   ├── incv.schema.json
│   ├── lsmixbc.schema.json
│   ├── none.schema.json
│   ├── nouterloop.schema.json
│   └── tstep4d.schema.json
├── aux
│   └── aux.schema.json
├── build
│   └── build.schema.json
├── dfi
│   └── dfi.schema.json
├── eda
│   └── eda.schema.json
├── geometry
│   ├── domain_name.schema.json
│   ├── domain.schema.json
│   ├── enum_for_nlon_nlat.jl
│   ├── geometry.schema.json
│   └── nlon_nlat.schema.json
├── main
│   ├── branches.json
│   ├── branches.schema.json
│   ├── date.schema.json
│   ├── emails.schema.json
│   ├── main.schema.json
│   ├── paths.schema.json
│   └── timelists.schema.json
├── nesting
│   ├── mars.schema.json
│   └── nesting.schema.json
├── observations
│   ├── liste_loc.schema.json
│   ├── observations.schema.json
│   └── odb
│       ├── codetype.schema.json
│       ├── ec2keyval.jq
│       ├── ecfilter.jq
│       ├── extract_from_ecmwf
│       ├── obstype2.json
│       ├── obstype.json
│       ├── obstype.schema.json
│       ├── reporttype.schema.json
│       └── varno.schema.json
├── physics
│   ├── alaro.schema.json
│   ├── arome.schema.json
│   ├── dynamics.schema.json
│   └── physics.schema.json
├── postprocessing
│   └── postprocessing.schema.json
├── surfex
│   ├── namelist
│   │   ├── nam_sson.schema.json
│   │   ├── nam_teb.schema.json
│   │   └── surfex_namelist.schema.json
│   └── surfex.schema.json
└── system
    ├── config
    │   └── ecgb-cca.json
    ├── config.schema.json
    ├── hostdescriptions.schema.json
    ├── linuxpc.json
    └── system.schema.json

17 directories, 57 files
roels@pc4523:/media/roels/_disk2/git/Harmonie.jl/docs/schema (master)$ 
```

Implementation

https://github.com/Hirlam/Harmonie.jl

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
JSON                            56             98              0           5062
TOML                           115            219            418           2273
YAML                             4             14              0            387
Julia                            7             71             29            169
HTML                             1             20             22             49
Markdown                         1             12              0             32
-------------------------------------------------------------------------------
SUM:                           184            434            469           7972
-------------------------------------------------------------------------------
```

Note 97% of SLOC is language independent JSON/TOML/YAML files.

HTML used for prepIFS like gui (see later slides)

Julia used to

- Convert TOML to config_exp.h format
- Unit-tests (next slide)
- Validate TOML files.

https://julialang.org/blog/2012/02/why-we-created-julia

Norwegian Meteorological Institute

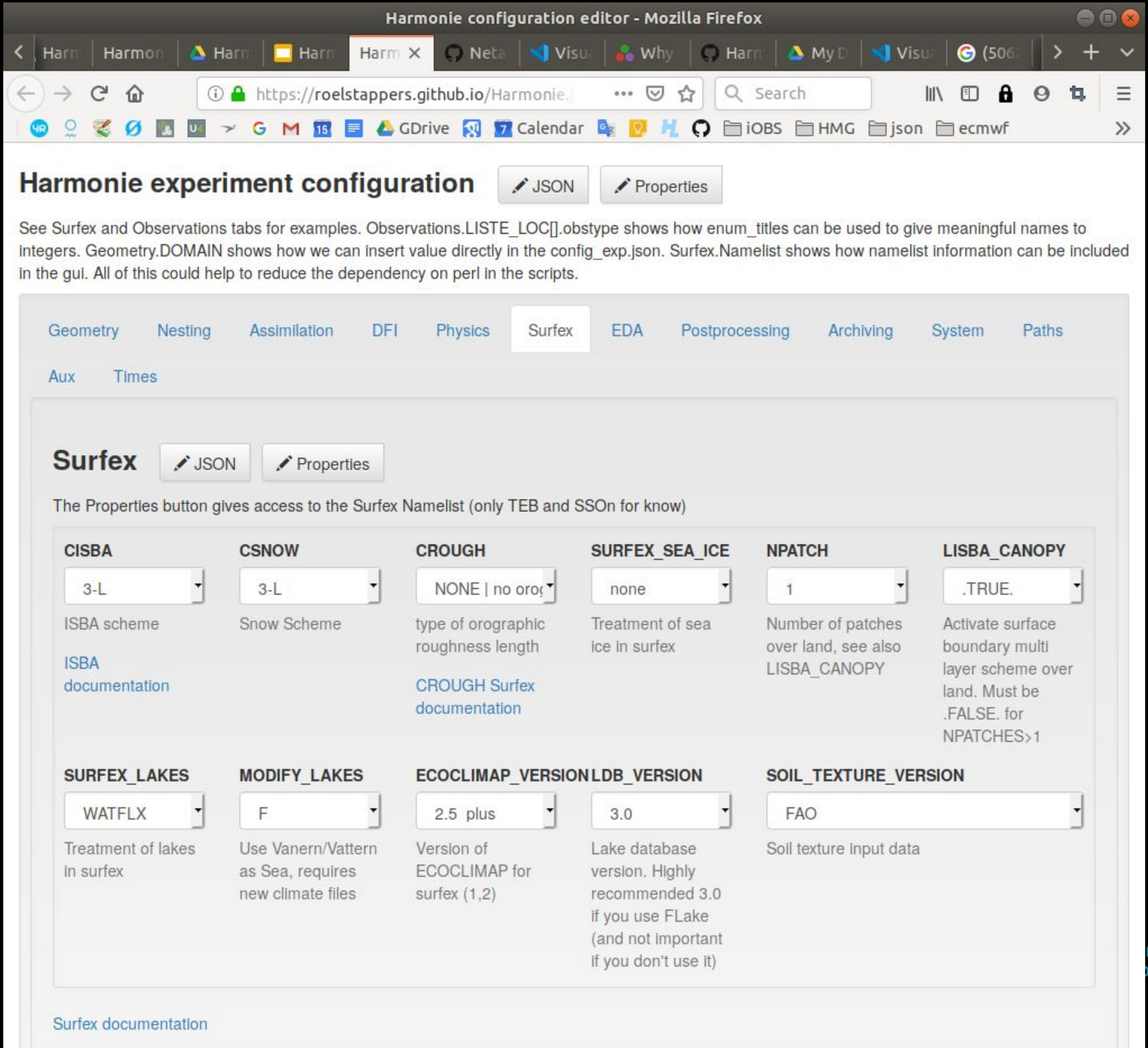
Unit tests

Any push to github automatically runs travis-CI to validate testbed configurations (work in progress).

Testbed configurations are created by merging toml files.

```
# AROME 4DVAR
@testset "AROME_4DVAR" begin
    config_exp   = TOML.parsefile("config/config_exp.toml")
    arome_4dvar = TOML.parsefile("harmonie_configurations/arome_4dvar.toml")
    config_exp_with_arome_4dvar = merge(merge, config_exp, arome_4dvar)
    @test Harmonie.isvalid(config_exp_with_arome_4dvar)
    merged_arome_4dvar = TOML.parsefile("testbed_configurations/arome_4dvar.toml")
    testbedconfig = merge(merge, config_exp_with_arome_4dvar, merged_arome_4dvar)
    @test Harmonie.isvalid(testbedconfig)
end
```

Harmonie GUI


Harmonie configuration editor - Mozilla Firefox
Harm
Harmon
Harm
Harm
Harm
Neta
Visu
Why
Harm
My D
Visu
(506
https://roelstappers.github.io/Harmonie
Search
GDrive
Calendar
iOBS
HMG
json
ecmwf

Harmonie experiment configuration

JSON | Properties

See Surfex and Observations tabs for examples. Observations.LISTE_LOC[].obstype shows how enum_titles can be used to give meaningful names to integers. Geometry.DOMAIN shows how we can insert value directly in the config_exp.json. Surfex.Namelist shows how namelist information can be included in the gui. All of this could help to reduce the dependency on perl in the scripts.

Geometry | Nesting | Assimilation | DFI | Physics | Surfex | EDA | Postprocessing | Archiving | System | Paths | Aux | Times

Surfex

JSON | Properties

The Properties button gives access to the Surfex Namelist (only TEB and SSOn for know)

CISBA: 3-L
ISBA scheme
ISBA documentation

CSNOW: 3-L
Snow Scheme

CROUGH: NONE | no oro(
type of orographic roughness length
CROUGH Surfex documentation

SURFEX_SEA_ICE: none
Treatment of sea ice in surfex

NPATCH: 1
Number of patches over land, see also LISBA_CANOPY

LISBA_CANOPY: .TRUE.
Activate surface boundary multi layer scheme over land. Must be .FALSE. for NPATCHES>1

SURFEX_LAKES: WATFLX
Treatment of lakes in surfex

MODIFY_LAKES: F
Use Vanern/Vattern as Sea, requires new climate files

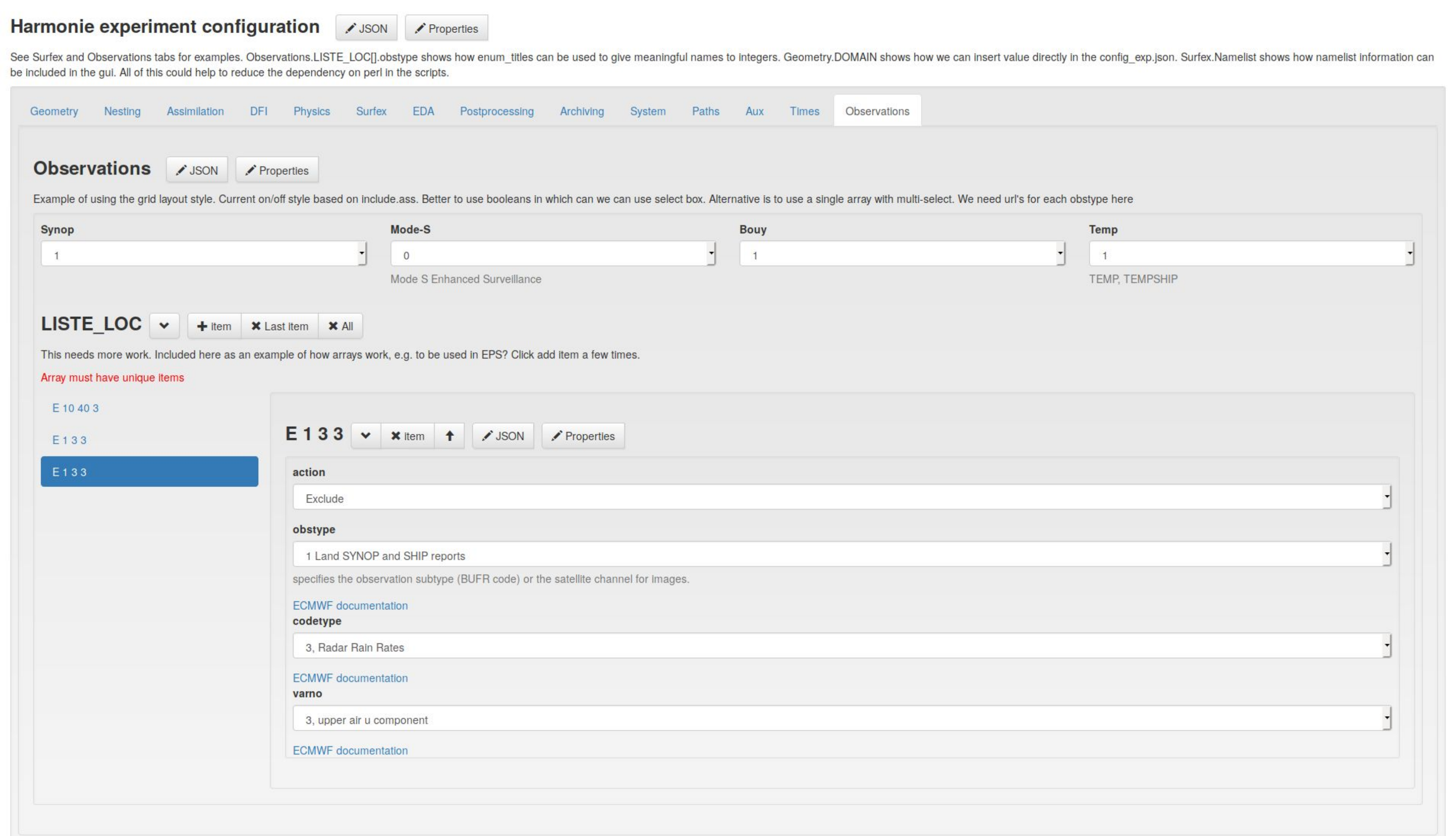
ECOCLIMAP_VERSION: 2.5_plus
Version of ECOCLIMAP for surfex (1,2)

LDB_VERSION: 3.0
Lake database version. Highly recommended 3.0 if you use FLake (and not important if you don't use it)

SOIL_TEXTURE_VERSION: FAO
Soil texture input data

Surfex documentation

LISTE_LOC example



Harmonie experiment configuration

See Surfex and Observations tabs for examples. Observations.LISTE_LOC[].obstype shows how enum_titles can be used to give meaningful names to integers. Geometry.DOMAIN shows how we can insert value directly in the config_exp.json. Surfex.Namelist shows how namelist information can be included in the gui. All of this could help to reduce the dependency on perl in the scripts.

Observations

Example of using the grid layout style. Current on/off style based on include.ass. Better to use booleans in which can we can use select box. Alternative is to use a single array with multi-select. We need url's for each obstype here

Synop: 1

Mode-S: 0
Mode S Enhanced Surveillance

Bouy: 1

Temp: 1
TEMP, TEMPSHIP

LISTE_LOC

This needs more work. Included here as an example of how arrays work, e.g. to be used in EPS? Click add item a few times.

Array must have unique items

E 1 3 3

action: Exclude

obstype: 1 Land SYNOP and SHIP reports
specifies the observation subtype (BUFR code) or the satellite channel for images.

ECMWF documentation

codetype: 3, Radar Rain Rates

ECMWF documentation

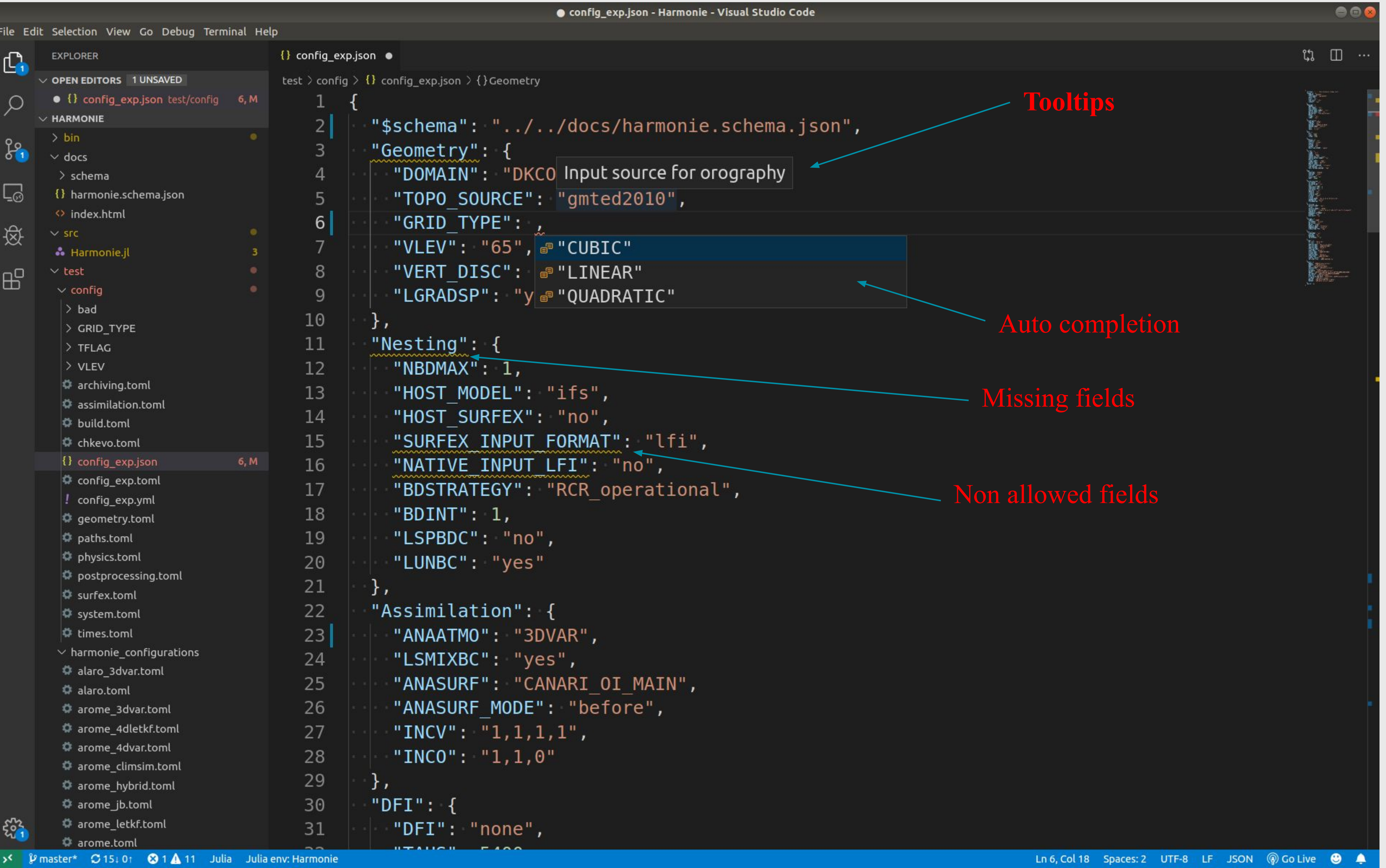
varno: 3, upper air u component

ECMWF documentation

GUI

- The GUI should remain lightweight, and low maintenance
- Use of the GUI should be optional.
- Relation between GUI and toml should be transparent
- GUI will be similar to prepIFS
- Need to a solution to handle the configuration for the linking of files in scripts (similar to Olive/Vortex)

Editor support (JSON/YAML only)



```
{
  "$schema": "../../docs/harmonie.schema.json",
  "Geometry": {
    "DOMAIN": "DKCO
    "TOPO_SOURCE": "gmted2010",
    "GRID_TYPE": ,
    "VLEV": "65",
    "VERT_DISC":
    "LGRADSP": "y
  },
  "Nesting": {
    "NBDMAX": 1,
    "HOST_MODEL": "ifs",
    "HOST_SURFEX": "no",
    "SURFEX_INPUT_FORMAT": "lfi",
    "NATIVE_INPUT_LFI": "no",
    "BDSTRATEGY": "RCR_operational",
    "BDINT": 1,
    "LSPBDC": "no",
    "LUNBC": "yes"
  },
  "Assimilation": {
    "ANAATMO": "3DVAR",
    "LSMIXBC": "yes",
    "ANASURF": "CANARI_OI_MAIN",
    "ANASURF_MODE": "before",
    "INCV": "1,1,1,1",
    "INCO": "1,1,0"
  },
  "DFI": {
    "DFI": "none",
```

Input source for orography

```
"CUBIC"
"LINEAR"
"QUADRATIC"
```

Tooltips

Auto completion

Missing fields

Non allowed fields

Code structure (repositories)

- In Harmonie scripts and configuration have always been in a single repository.
- Currently Harmonie.jl is a separate repository with configuration for config_exp.toml. This is similar to how prepIFS is used at ECMWF. Is it preferable to have a single repository with scripts and configuration or are there benefits to split this in two repositories ?
- Automatic generation of scripts (like Olive/Vortex)?

Summary/To do list

A prototype for the handling and validating configuration data has been presented.

- Based on language independent format (TOML/YAML/JSON)
- Validation using JSON-Schema
- Works in Harmonie `/perm/ms/no/fars/worktrees/jsonschema`
- Testbed runs successfully
- Configuration now in a separate git-repository

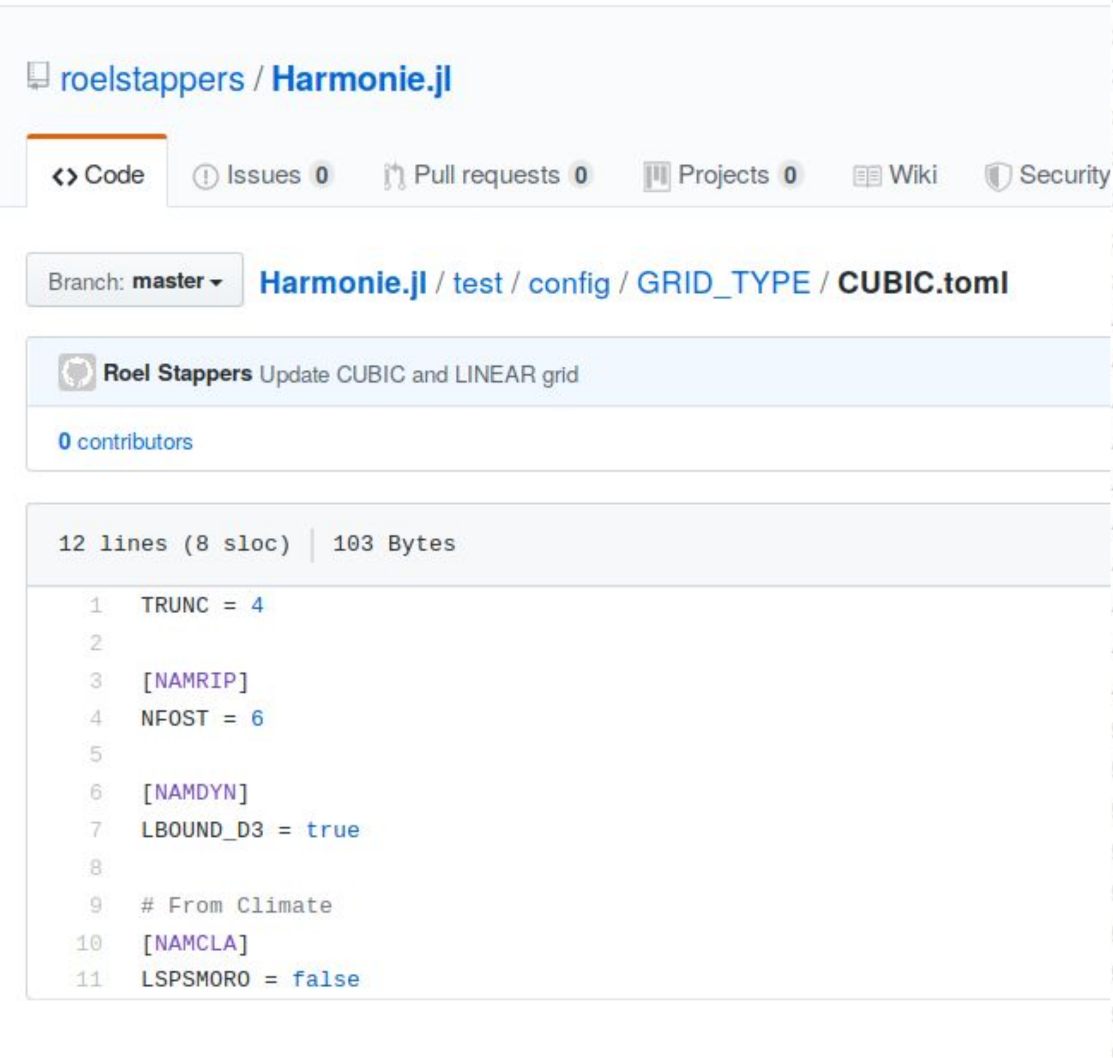
To do:

- Needs further extension for 1) namelists, 2) param_bator.cfg etc., 3) LISTE_LOC, LISTE_NOIRE_DIAP, 4) codetype obstype, 5) submit.ecgb-cca, submit.LinuxPC, 6) job submission, 7) compiler options, 8) VarBC predictors, 9) MARS request files, 10) ECMWF cycles, 11) ecflow tasks/families 12) testbed
- Grouping of variables.
- Scripts should start extracting from config_exp.toml directly without needing the `export` statements in config_exp.h

Script simplification

- Use TOML config to move “business logic” out of scripts

E.g. in Climate



roelstappers / **Harmonie.jl**

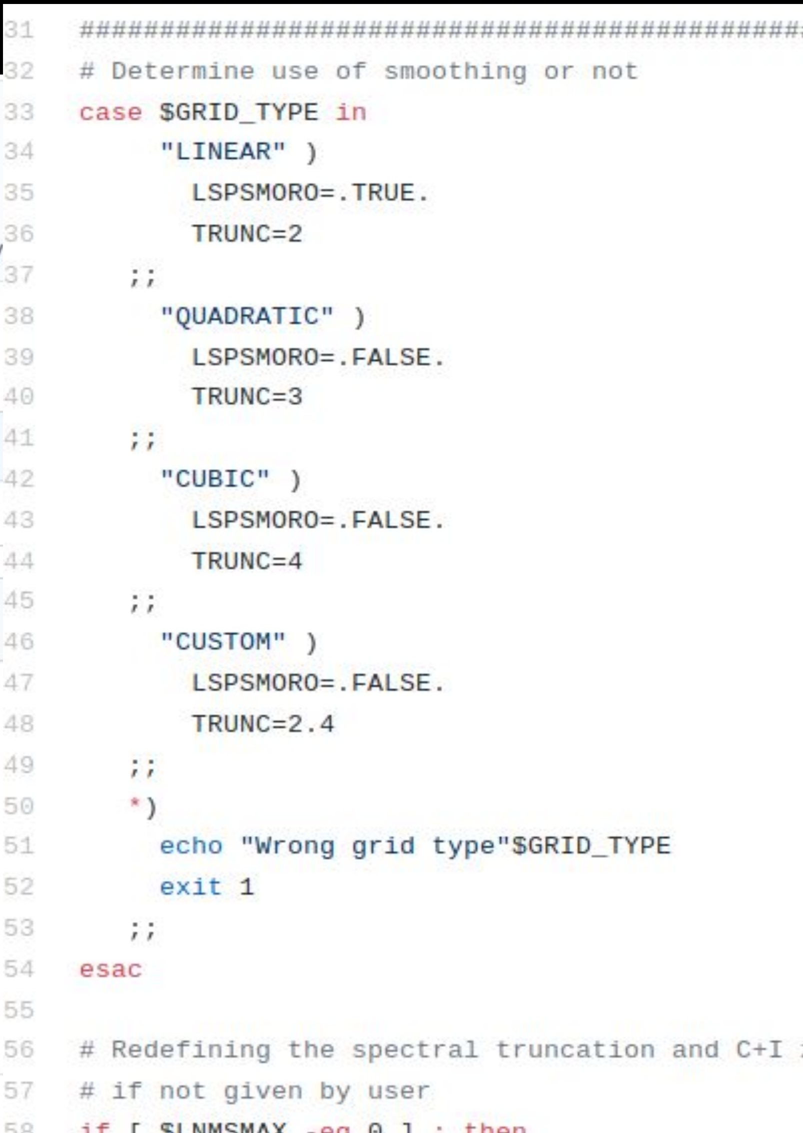
Harmonie.jl / test / config / GRID_TYPE / **CUBIC.toml**

Roel Stappers Update CUBIC and LINEAR grid

0 contributors

12 lines (8 sloc) | 103 Bytes

```toml
TRUNC = 4

[NAMRIP]
NFOST = 6

[NAMDYN]
LBOUND_D3 = true

# From Climate
[NAMCLA]
LSPSMORO = false
```



```bash
##############################################
# Determine use of smoothing or not
case $GRID_TYPE in
     "LINEAR" )
       LSPSMORO=.TRUE.
       TRUNC=2
    ;;
     "QUADRATIC" )
       LSPSMORO=.FALSE.
       TRUNC=3
    ;;
     "CUBIC" )
       LSPSMORO=.FALSE.
       TRUNC=4
    ;;
     "CUSTOM" )
       LSPSMORO=.FALSE.
       TRUNC=2.4
    ;;
    *)
     echo "Wrong grid type"$GRID_TYPE
     exit 1
    ;;
esac

# Redefining the spectral truncation and C+I
# if not given by user
if [ $LNMSMAX -eq 0 ] ; then
```

default values in Namelist schema’s

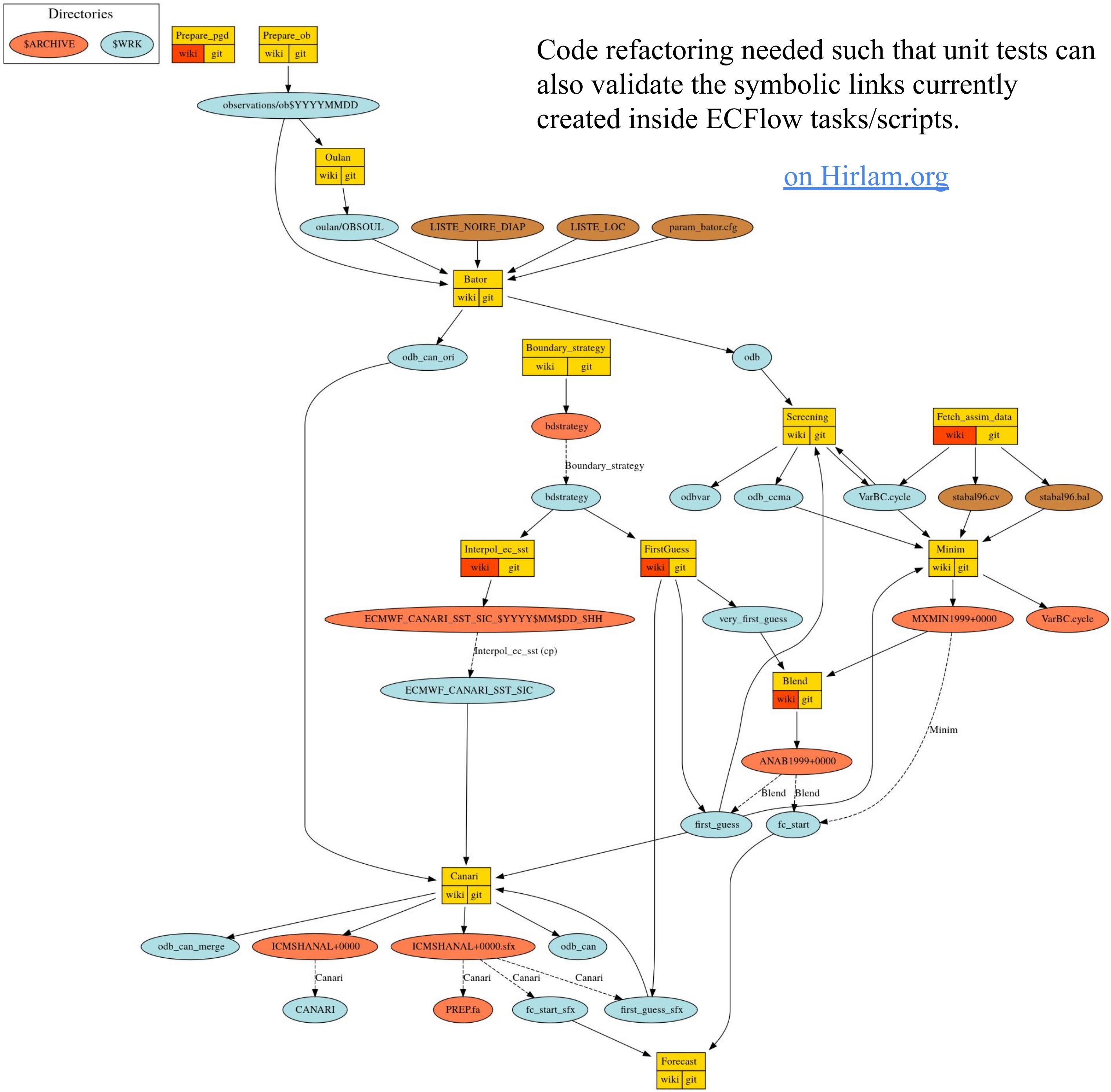
JSON-schema allows specification of default values. They are not part of the validation but tools can use this to fill in missing values.

For namelists this would mean we have to pick default values consistent with the IFS/surfex code.

Which would force the use of the default LELAM=false

However we can have `enum : [ true ]` such that not explicitly setting specifying LELAM=false is invalid.

Data Assimilation in Harmonie-Arome

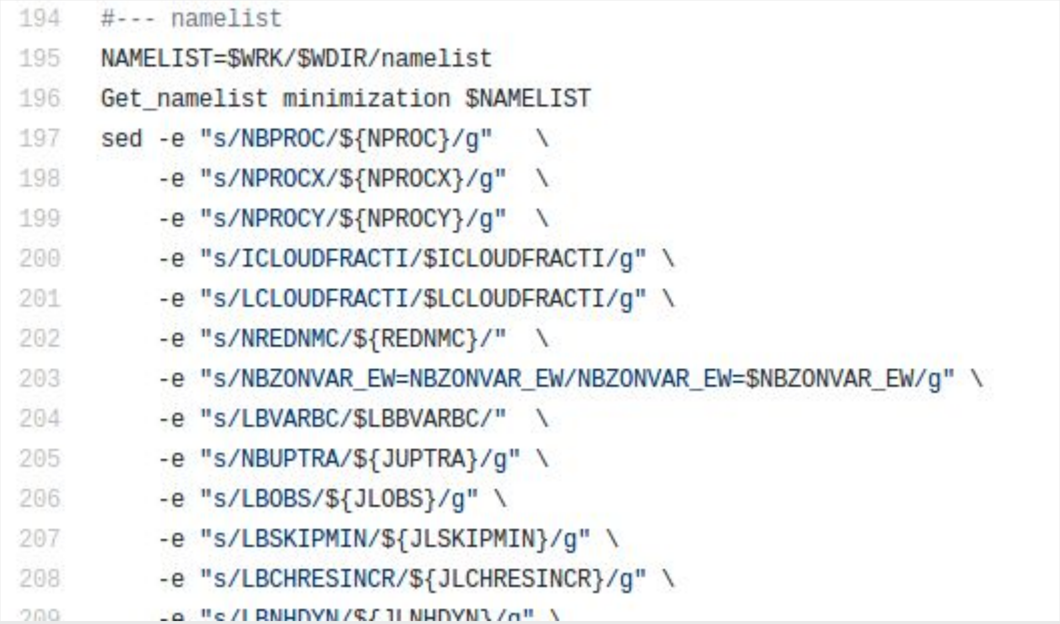


Code refactoring needed such that unit tests can also validate the symbolic links currently created inside ECFlow tasks/scripts.

on Hirlam.org

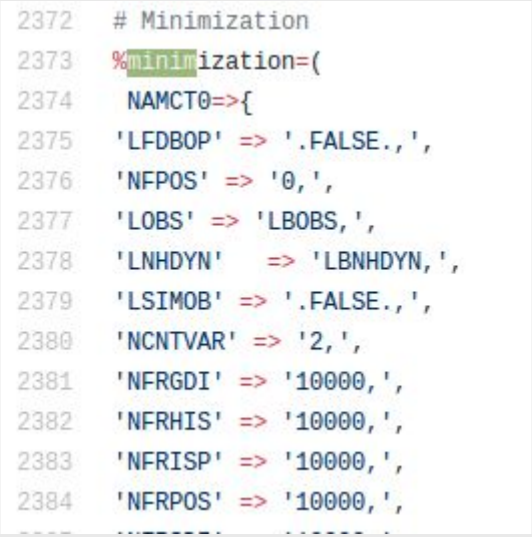
TOML for Namelists

scr/Mimim

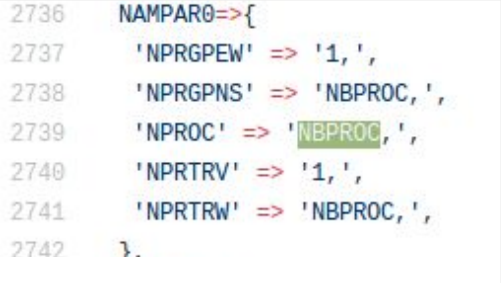


```
#--- namelist
NAMELIST=$WRK/$WDIR/namelist
Get_namelist minimization $NAMELIST
sed -e "s/NBPROC/${NPROC}/g"   \
    -e "s/NPROCX/${NPROCX}/g"  \
    -e "s/NPROCY/${NPROCY}/g"  \
    -e "s/ICLOUDFRACTI/$ICLOUDFRACTI/g" \
    -e "s/LCLOUDFRACTI/$LCLOUDFRACTI/g" \
    -e "s/NREDNMC/${REDNMC}/"  \
    -e "s/NBZONVAR_EW=NBZONVAR_EW/NBZONVAR_EW=$NBZONVAR_EW/g" \
    -e "s/LBVARBC/$LBBVARBC/"  \
    -e "s/NBUPTRA/${JUPTRA}/g" \
    -e "s/LBOBS/${JLOBS}/g" \
    -e "s/LBSKIPMIN/${JLSKIPMIN}/g" \
    -e "s/LBCHRESINCR/${JLCHRESINCR}/g" \
    -e "s/LBNHDYN/${JLNHDYN}/g" \
```

nam/Harmonie_namelist.pm

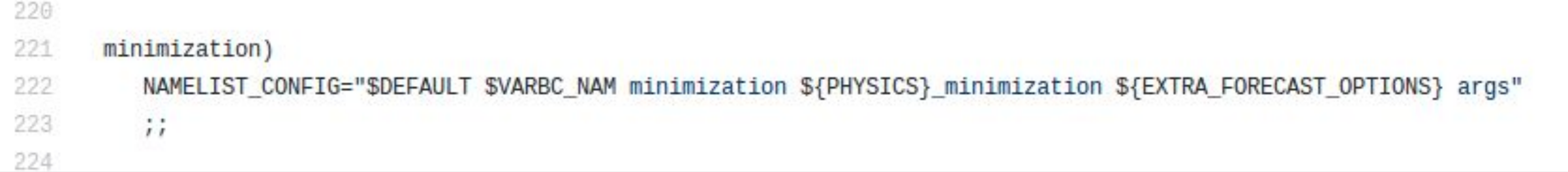


```
# Minimization
%minimization=(
 NAMCT0=>{
'LFDBOP' => '.FALSE.,',
'NFPOS' => '0,',
'LOBS' => 'LBOBS,',
'LNHDYN'    => 'LBNHDYN,',
'LSIMOB' => '.FALSE.,',
'NCNTVAR' => '2,',
'NFRGDI' => '10000,',
'NFRHIS' => '10000,',
'NFRISP' => '10000,',
'NFRPOS' => '10000,',
```

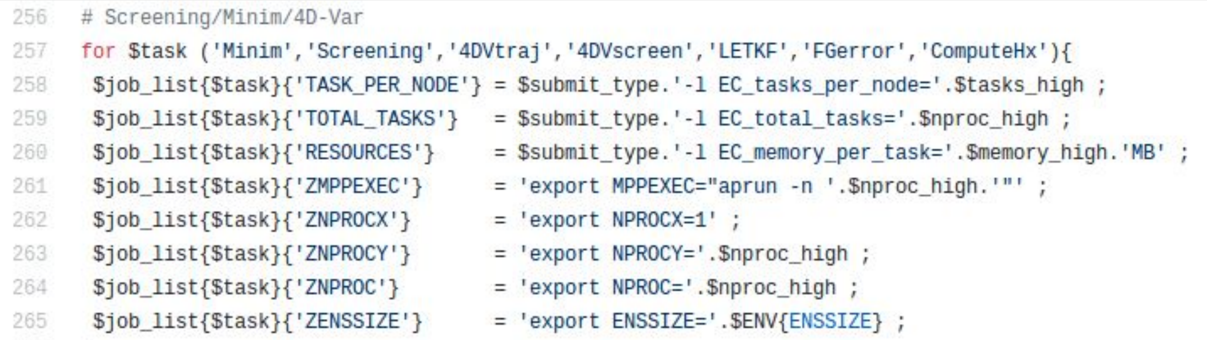


```
 NAMPAR0=>{
  'NPRGPEW' => '1,',
  'NPRGPNS' => 'NBPROC,',
  'NPROC' => 'NBPROC,',
  'NPRTRV' => '1,',
  'NPRTRW' => 'NBPROC,',
 },
```

scr/Get_namelist



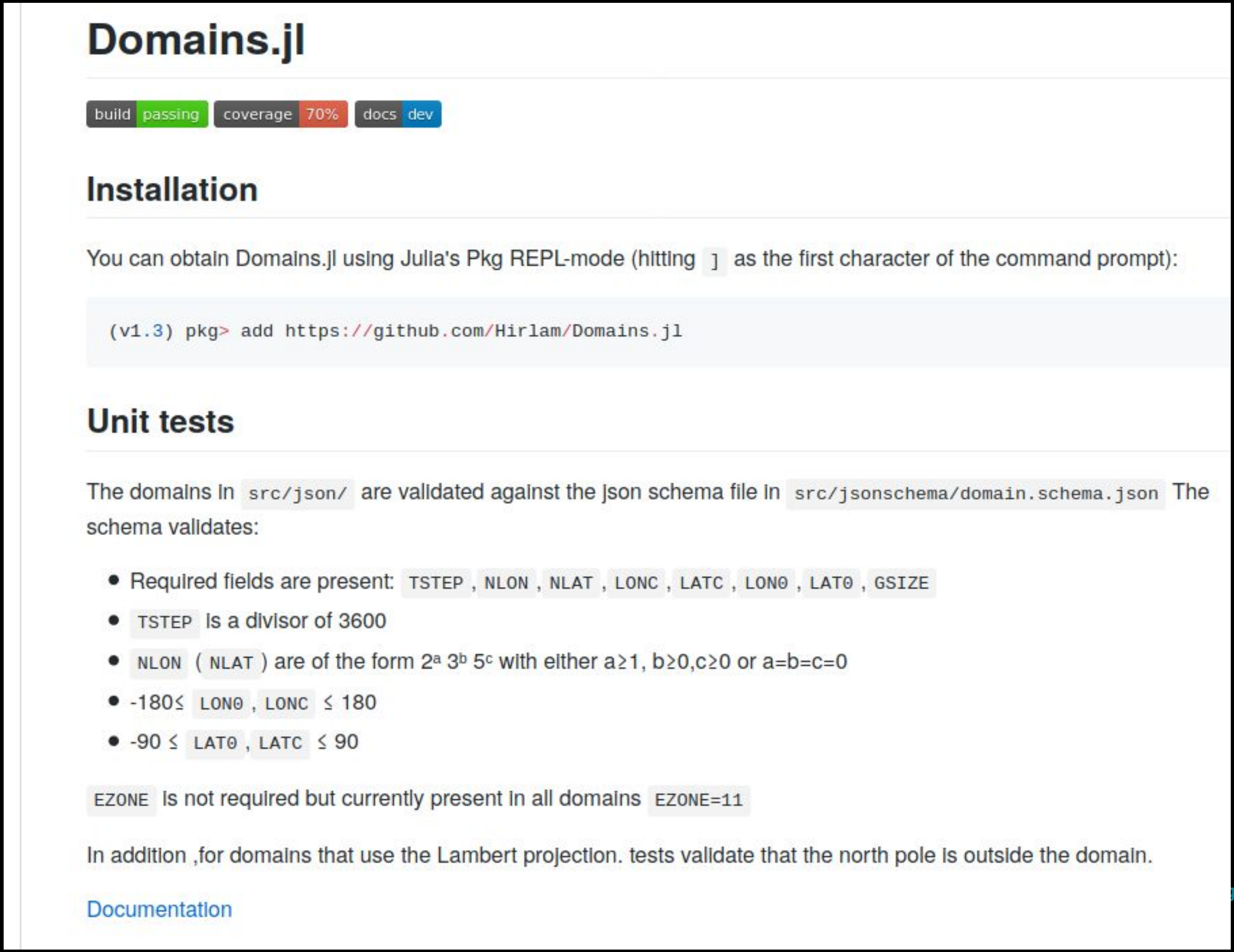
```

  minimization)
    NAMELIST_CONFIG="$DEFAULT $VARBC_NAM minimization ${PHYSICS}_minimization ${EXTRA_FORECAST_OPTIONS} args"
    ;;

```



```
# Screening/Minim/4D-Var
for $task ('Minim','Screening','4DVtraj','4DVscreen','LETKF','FGerror','ComputeHx'){
 $job_list{$task}{'TASK_PER_NODE'} = $submit_type.'-l EC_tasks_per_node='.$tasks_high ;
 $job_list{$task}{'TOTAL_TASKS'}   = $submit_type.'-l EC_total_tasks='.$nproc_high ;
 $job_list{$task}{'RESOURCES'}     = $submit_type.'-l EC_memory_per_task='.$memory_high.'MB' ;
 $job_list{$task}{'ZMPPEXEC'}      = 'export MPPEXEC="aprun -n '.$nproc_high.'"' ;
 $job_list{$task}{'ZNPROCX'}       = 'export NPROCX=1' ;
 $job_list{$task}{'ZNPROCY'}       = 'export NPROCY='.$nproc_high ;
 $job_list{$task}{'ZNPROC'}        = 'export NPROC='.$nproc_high ;
 $job_list{$task}{'ZENSSIZE'}      = 'export ENSSIZE='.$ENV{ENSSIZE} ;
```

Norwegian
Meteorological
Institute

Domains



Domains.jl

build passing coverage 70% docs dev

Installation

You can obtain Domains.jl using Julia's Pkg REPL-mode (hitting `]` as the first character of the command prompt):

```
(v1.3) pkg> add https://github.com/Hirlam/Domains.jl
```

Unit tests

The domains in `src/json/` are validated against the json schema file in `src/jsonschema/domain.schema.json` The schema validates:

- Required fields are present: `TSTEP` , `NLON` , `NLAT` , `LONC` , `LATC` , `LON0` , `LAT0` , `GSIZE`
- `TSTEP` is a divisor of 3600
- `NLON` (`NLAT`) are of the form $2^a 3^b 5^c$ with either a≥1, b≥0,c≥0 or a=b=c=0
- -180≤ `LON0` , `LONC` ≤ 180
- -90 ≤ `LAT0` , `LATC` ≤ 90

`EZONE` is not required but currently present in all domains `EZONE=11`

In addition ,for domains that use the Lambert projection. tests validate that the north pole is outside the domain.

Documentation