Unified Workflow Tools: An Agile Approach to Developing a Configuration Toolbox

Naureen Bharwani[1,2], **Emily Carpenter**[1,2], Christina Holt[1,2], Paul Madden[1,2], Fredrick Gabelmann[5,6], Brian Weir[6,7]

[1]Cooperative Institute for Research in Environmental Sciences (CIRES), [2]National Oceanic and Atmospheric Administration, Global Systems Laboratory (NOAA GSL), [5]Element 84, Inc., [6]NOAA Earth Prediction Innovation Center (EPIC), [7]RTX Corporation.

Background

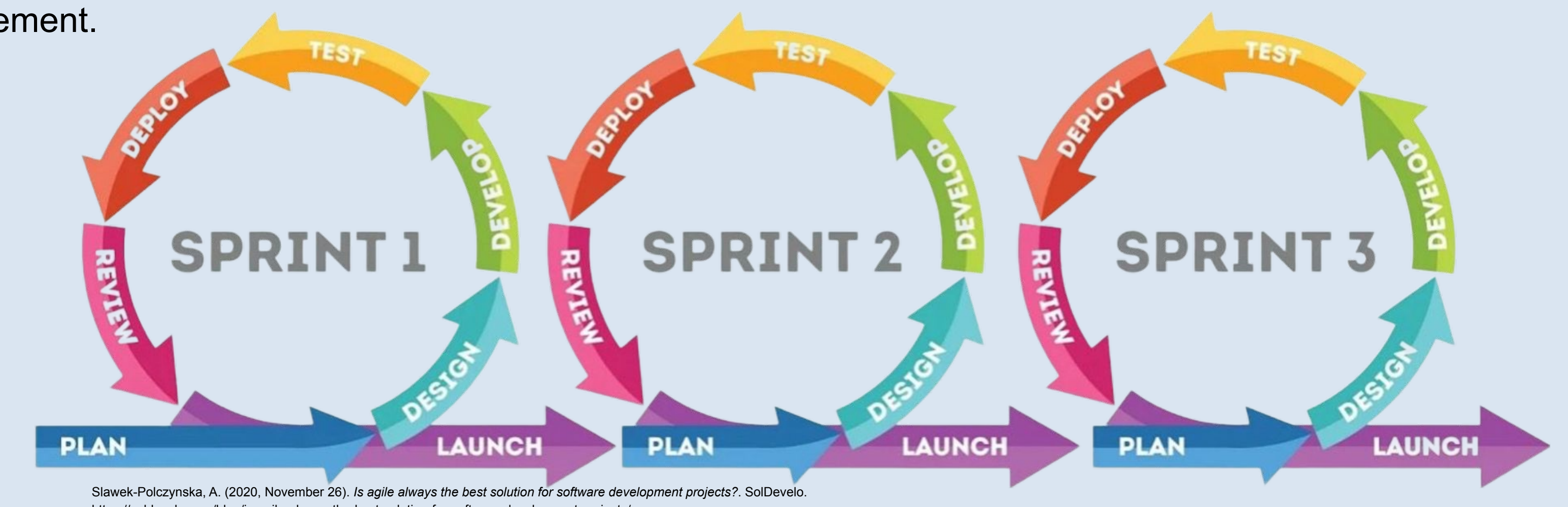
The **Unified Forecast System (UFS)** is a community-based Earth modeling system comprising data assimilation, numerical weather prediction, post-processing components and many others. It promotes a collaborative environment to support research and improve operational outcomes through increased forecast accuracy.

The **Earth Prediction Innovation Center (EPIC)** aims to improve the UFS by modernizing software practices, leveraging cutting-edge cloud technologies, and optimizing the utilization of high-performance computing resources. As a pioneering initiative within the **National Oceanic and Atmospheric Administration (NOAA)** and in collaboration with Raytheon Intelligence & Space, EPIC drives advancements in weather forecasting and Earth system modeling, facilitating collaborative research among interdisciplinary teams.

Historically, each UFS application (e.g. Short Range Weather) has implemented bespoke tools and configuration schemes to drive its workflows, leading to duplication of effort and lack of compatibility across UFS apps. The **Unified Workflow (UW)** team is one of several teams that compose EPIC and consists of NOAA **Global Systems Laboratory** and EPIC staff. The UW team is developing a set of reusable, interoperable tools to harmonize solutions for needs common to all apps.

Agile

EPIC integrates a project management system called **Scaled Agile Framework**, which guides organizations through the agile process. **Agile** is a set of practices that enable teams to navigate complex projects by breaking them into manageable increments ("sprints"), emphasizing iterative development, and promoting continuous improvement.



Slawek-Polczynska, A. (2020, November 26). *Is agile always the best solution for software development projects?*. SolDevelo. https://soldevelo.com/blog/is-agile-always-the-best-solution-for-software-development-projects/

After each sprint, the UW team engages in a "retrospective", where we adapt our plan for the next sprint based on our experience and on feedback received from our stakeholders and community. Other EPIC teams simultaneously work toward complementary objectives as we each make progress toward our common goals.

Get Involved

We need your input! Agile works best when we receive feedback from the community. This is where you come in. Tell us what's going well, what needs improvement, and what new features you'd like to see. You can participate in our Agile development cycle by contributing to UW Tools at https://github.com/ufs-community/uwtools or by using the QR code.

uwtools on GitHub:

To learn more about UW Tools, you can checkout our documentation at https://uwtools.readthedocs.io/ or by using the QR code.

uwtools Documentation:

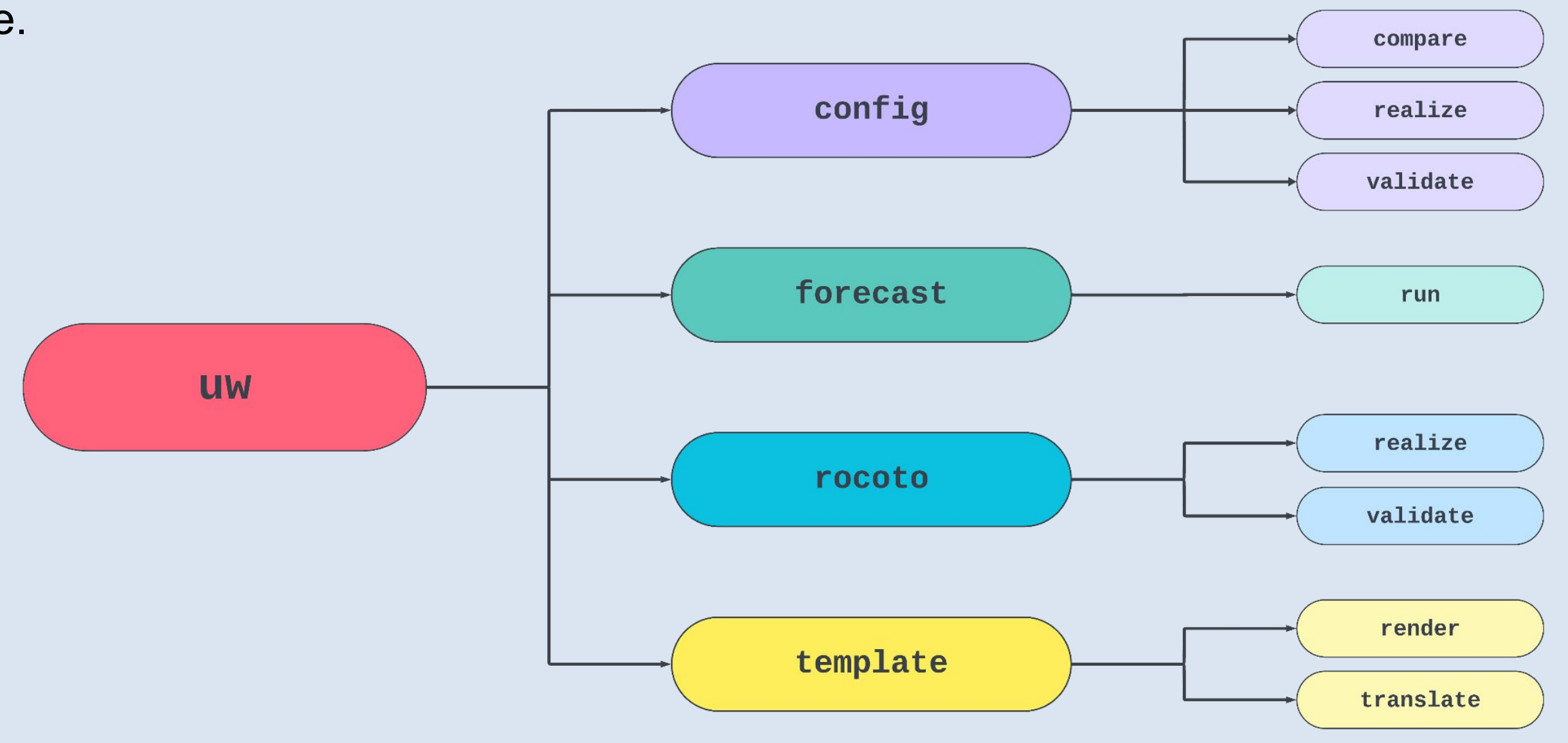
Our colleagues are speaking on UW Tools at AMS:

- Unified Workflow Tools and Framework: An Update by Christina Holt
- Unifying Workflows with the Strangler Fig Pattern by Brian Weir

Toolbox

Current Structure of the Unified Workflow Toolbox

The diagram below shows our current tools – **`config`**, **`forecast`**, **`rocoto`**, and **`template`** – accessed by invoking the **`uw`** program with a tool name and an operation. For example, configuration files can be compared by running **`uw config compare`** along with file names and options. See the *In Action:* ***`compare`*** section for a concrete example.



In Action: `config`

The **`config`** tool's **`realize`** operation transforms a "base" configuration into a fully-formed, app-ready file, guided by user-supplied specifications. It operates on configurations formatted as YAML, INI, Fortran namelist, or bash **`key=value`** pairs, and prepares the output by resolving references, computing values, formatting strings, and building lists.

Consider the transformation performed by the invocation, on the following files, of the command

`uw config realize --input-file base.yaml --values-file values.yaml`

`base.yaml`

```
platform:
   NCORES_PER_NODE: 40

task_run_fcst:
   RUN_FCST_TN: "run_fcst"
   PE_MEMBER01: 221
   NNODES_RUN_FCST: '{{ (PE_MEMBER01 + PPN_RUN_FCST - 1) // PPN_RUN_FCST }}'
   PPN_RUN_FCST: '{{ platform.NCORES_PER_NODE // OMP_NUM_THREADS_RUN_FCST }}'
   WTIME_RUN_FCST: 04:30:00
   MAXTRIES_RUN_FCST: 1
   OMP_NUM_THREADS_RUN_FCST: 2
```

`values.yaml`

```
task_run_fcst:
   OMP_NUM_THREADS_RUN_FCST: 4
```

output

```
platform:
   NCORES_PER_NODE: 40

task_run_fcst:
   RUN_FCST_TN: "run_fcst"
   NNODES_RUN_FCST: 12
   PPN_RUN_FCST: 20
   WTIME_RUN_FCST: 04:30:00
   MAXTRIES_RUN_FCST: 1
   OMP_NUM_THREADS_RUN_FCST: 4
```

Tell Us Your Pain Points!

Tell us about your biggest challenges or what tools you would like to see!

In Action: `template`

The Template tool renders a Jinja2 template with user-supplied values, which may be supplied via environment variables, YAML, INI, Fortran namelists, or bash **`key=value`** pairs.

For example, the file **`template`** can be rendered using values from the Fortran namelist file **`values.nml`**, as well as from environment variables, which would override the same-named namelist variable, as is shown here with the **`hh`** variable.

`template`

```
Forecast Report for Cycle {{ yyyymmdd }} {{ hh }}Z
```

`values.nml`

```
&fcst
     yyyymmdd = "20240111"
     hh = "18"
/
```

```
$ hh=00 uw template render --input-file template --values-file values.yaml
 Forecast Report for Cycle 20240111 00Z
```

The template can also be queried to find out which values need to be supplied for rendering:

```
$ uw template render --input-file template --values-needed
[2024-01-08T09:15:24]     INFO Value(s) needed to render this template are:
[2024-01-08T09:15:24]     INFO yyyymmdd
[2024-01-08T09:15:24]     INFO hh
```

In Action: `compare`

Have you ever found the Linux diff tool just doesn't work for comparing unordered namelists with mixed-case keys? Our **`uw config compare`** solves this very problem!

Here's an example with a Fortran namelist input and configuration files. Using **`uw config realize`**, you'd get the following output:

`base.nml`

```
 &fv_core_nml
   layout = 3, 8
   io_layout = 1,1
   npx = 97
   npy = 97
   ntiles = 6
   npz = 64
   grid_type = -1
   nudge_qv = .true.
   nudge_dz = .false.
   tau = 10.
   rf_cutoff = 7.5e2
   d2_bg_k1 = 0.20
   d2_bg_k2 = 0.0
   kord_tm = -9
   kord_mt = 9
   kord_wz = 9
   kord_tr = 9
   hydrostatic = .false.
   phys_hydrostatic = .false.
   use_hydro_pressure = .false.
 /
```

`values.nml`

```
 &fv_core_nml
   ntiles = 6
   npz = 64
   grid_type = -1
   layout = 3, 8
   npy = 97
   npx = 97
   nudge_qv = .true.
   npz = 65
   kord_wz = 9
   nudge_dz = .false.
   rf_cutoff = 7.5e2
   use_hydro_pressure = .true.
   d2_bg_k1 = 0.20
   d2_bg_k2 = 0.0
   tau = 10.
   kord_mt = 9
   hydrostatic = .false.
   phys_hydrostatic = .false.
   kord_tr = 9
 /
```

output

```
 &fv_core_nml
   layout = 3, 8
   io_layout = 1,1
   npx = 97
   npy = 97
   ntiles = 6
   npz = 65
   grid_type = -1
   nudge_qv = .true.
   nudge_dz = .false.
   tau = 10.
   rf_cutoff = 7.5e2
   d2_bg_k1 = 0.20
   d2_bg_k2 = 0.0
   kord_tm = -9
   kord_mt = 9
   kord_wz = 9
   kord_tr = 9
   hydrostatic = .false.
   phys_hydrostatic = .false.
   use_hydro_pressure = .true.
 /
```

To easily see the differences between these files, you can use **`uw config compare`**:

```
$ uw config compare --file-1-path base.nml --file-2-path values.nml
[2024-01-08T14:57:07]     INFO - base.nml
[2024-01-08T14:57:07]     INFO + values.nml
[2024-01-08T14:57:07]     INFO ------------------------------------------------------------------
[2024-01-08T14:57:07]     INFO fv_core_nml:                      npz: - 64 + 65
[2024-01-08T14:57:07]     INFO fv_core_nml:                  kord_tm: - -9 + None
[2024-01-08T14:57:07]     INFO fv_core_nml:       use_hydro_pressure: - False + True
```

This research was supported in part by NOAA cooperative agreement NA22OAR4320151, an FY22 WPO JTTI award, the OCIO Software Engineering for Novel Architectures program, and the EPIC Contract.