IFS*hub*

An Integrated workflow interface for ECMWF

Paul Burton

Paolo Battino, Sylvie Lamy-Thepaut, Eduard Rosert, Krzysztof Sciubisz

ECMWF

Paul.Burton@ecmwf.int

ECMWF

Overview

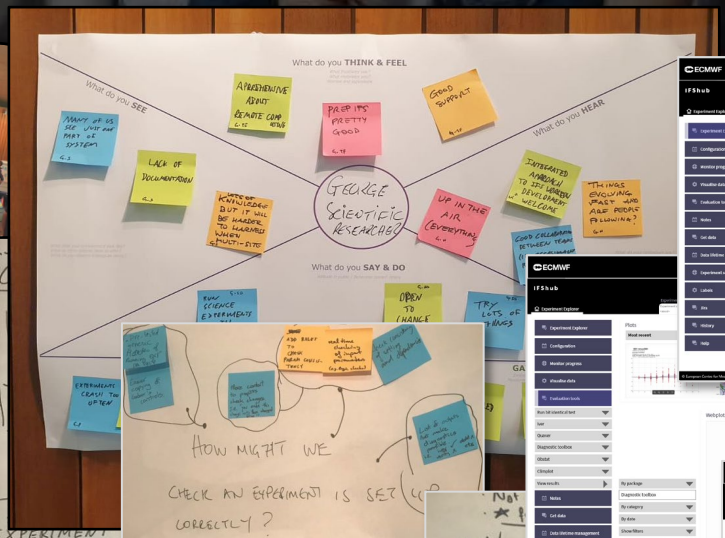
- Motivation & vision for change
- IFS*hub* in a nutshell
- Technical implementation
- *web*PrepIFS
- Future opportunities

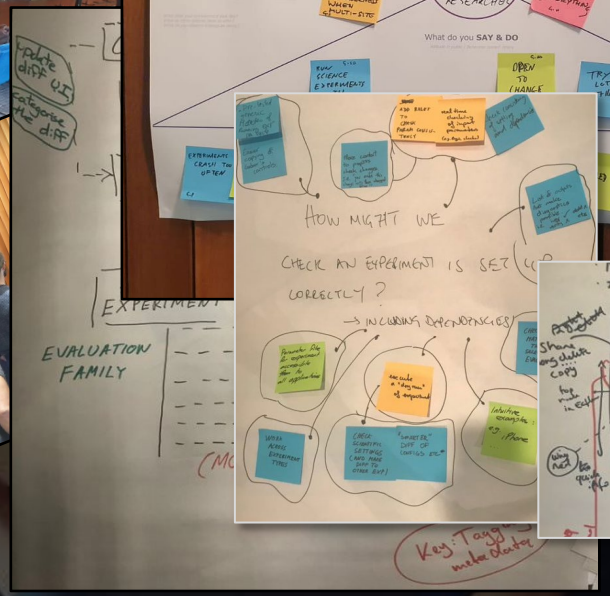
Motivations

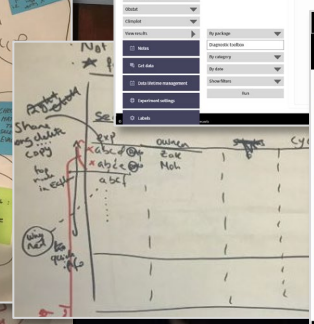
- IFS experiment configuration tool “prepIFS” needs replacing
  - 25 year old Java application, aging UI design
  - Difficult to introduce new functionality
  - Want a user friendly, flexible framework for active development
- Want a seamless management of developer workflow
  - Currently we use different tools on different platforms with no direct integration
  - Link experiment resources together (configuration / metadata / data / plots / data governence)
  - Lower the learning threshold for new staff
- Better support for distributed / remote working
  - Without requiring additional software installation or high bandwidth / low latency connections
- Insulate against IT infrastructure changes
- Easy integration & use of remote resources
  - Destination Earth

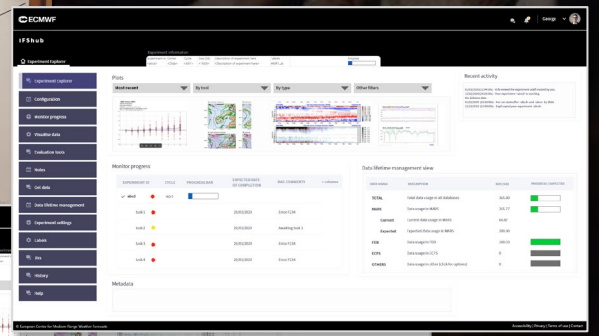
Developing a vision

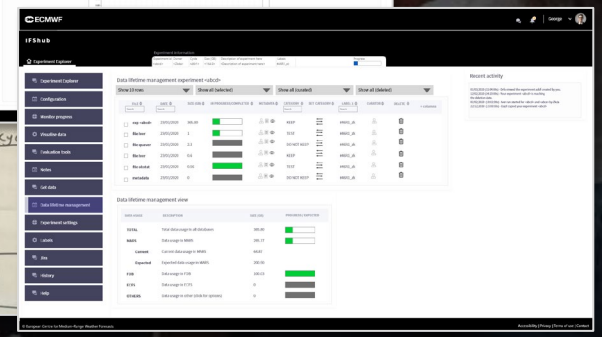
1. Identify users & roles
2. Understand workflow & pain points
3. Brainstorm solutions
4. Wireframes & feedback







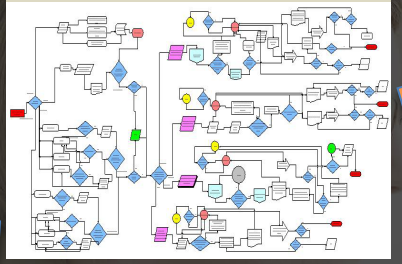




IFS Developer Workflow

configure

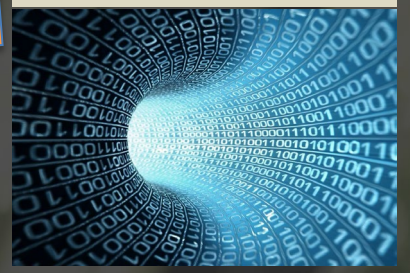
monitor

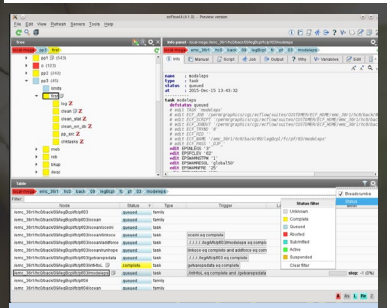


analyse

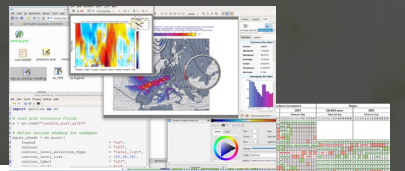
collaborate

manage

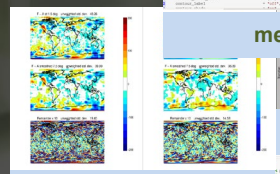




ecFlowUI



metview



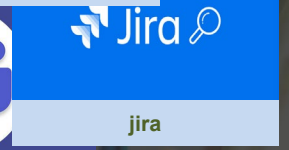
iver

quaver

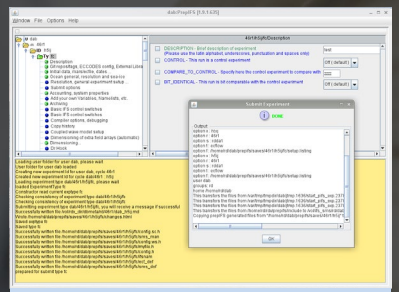
jupyter

confluence

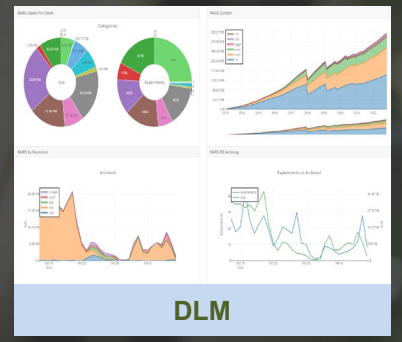
teams



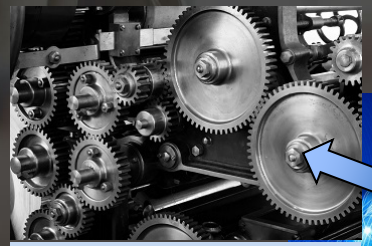
jira



prepIFS

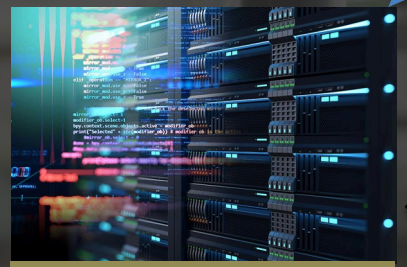


DLM

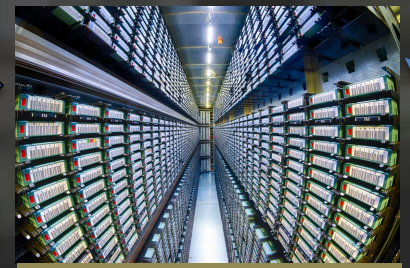


ecFlow Server

HPC



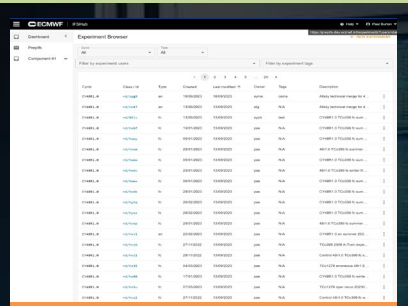
filesystems



archive

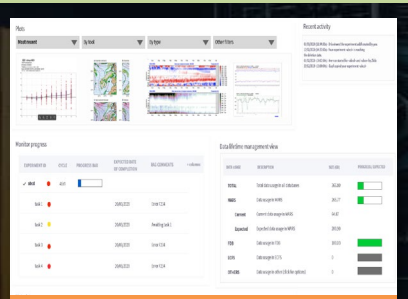
IFS*hub* Workflow

discover



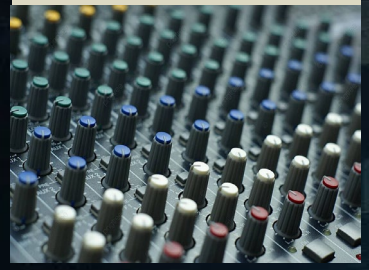
experiment discovery

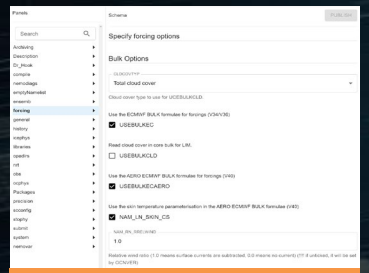
mission control



experiment explorer

configure

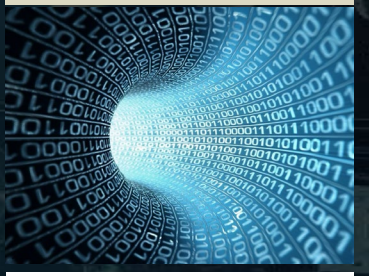


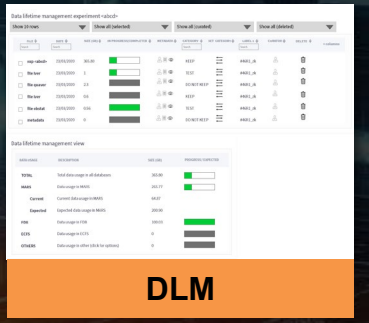


*web*PrepIFS

IFS*hub*

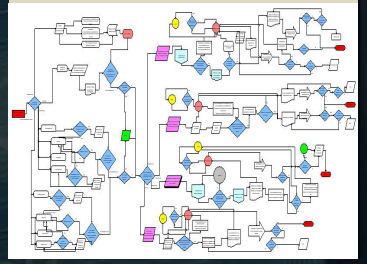
manage

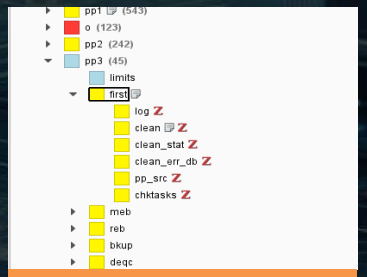




DLM

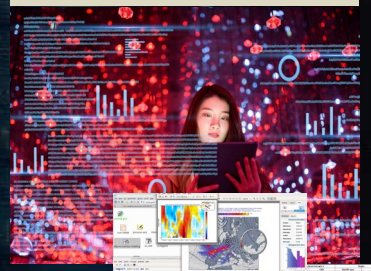
monitor



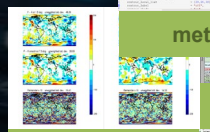


ecFlow status

analyse



metview



quaver

iver

jupyter

collaborate

confluence

teams

jira

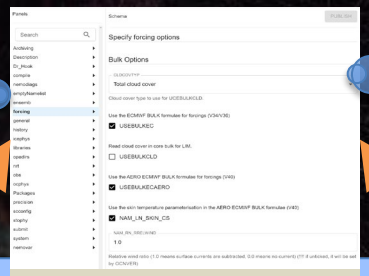
IFS*hub* System Architecture

Built using Typescript & React

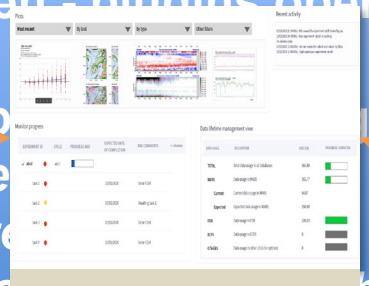
plugins mounted as iframe

IFS*hub* frontend

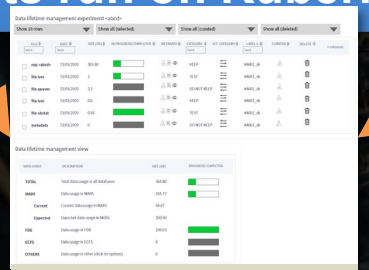
IFS*hub* plugin controller



plugin #1 frontend



plugin #2 frontend



plugin #3 frontend

- Standalone application
- No restrictions on technology
- Subscribes to IFS*hub*

ECMWF identity provider

- First layer of user authentication
- Connects to ECMWF identity provider & Active Directory

plugin #1 backend

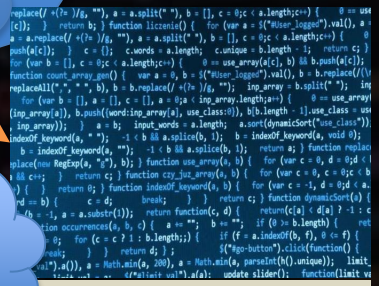
authentication

Python & Django framework

plugin #2 backend

- Standalone service exposing http API for IFS*hub* backend to consume
- No restrictions on technology

IFS*hub* backend



plugin #3 backend

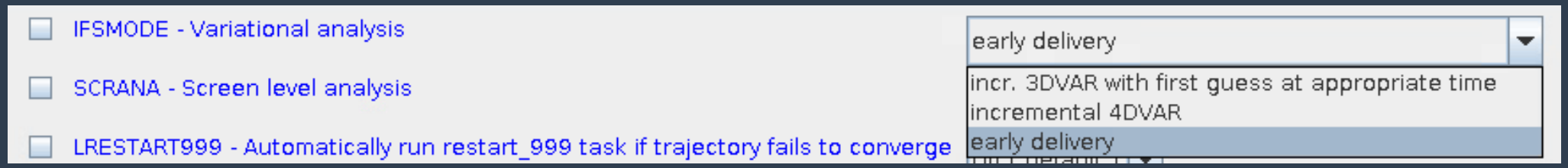
Access to database via IFS*hub* backend

Controller for plugin applications *(subscribe/publish)*

*web*PrepIFS

- Initial Implementation
  - Focus on reproducing functionality of existing desktop version
  - But improving usability based on user feedback & pain points
- Flexible user-defined configuration UI
  - Users can modify the schema to modify/add/remove variables
    - Each experiment links to a specific schema defined in a git repository
      - Standard schema for each IFS release, but users can write their own branches to modify it
  - Basic internal rule language
    - Checking for consistent/correct values
    - Automatic modification of variables
      - e.g. setting HPC configuration tasks/threads based on resolution
  - Generates configuration files in various formats
  - Aiming for a generic system – user-defined schema defines UI & behaviour
    - Not restricted to IFS or ECMWF systems

Example of *web*PrepIFS JSON schema



- Basic description of variable
  - name/type & default value
  - GUI options on display
  - GUI options on input field
    - Here we have a “selectone” drop-down menu with 3 choices
    - Each choice has some descriptive text, and a value which will be written to configuration file(s)
  - Output to the selected files – format defined by file_type
- User simply modifies a file and commits it to update the GUI for ther experiment

```
{
  "variable": {
    "name": "IFSMODE",
    "type": "string",
    "default": "early_delivery",
    "gui": {
      "label": {"en-GB": "Variational analysis"},
      "help": {"en-GB": "incremental 3DVAR with first guess at appropriate time"},
      "widget": {
        "select": {
          "selectone": "drop-down",
          "choices": [
            {"3d_fgat": "incr. 3DVAR with first guess at appropriate time"},
            {"4d_inc": "incremental 4DVAR"},
            {"early_delivery": "early delivery"}
          ]
        }
      },
    },
    "output": {
      "files": {
        "ecf_def": {"file_type": "ecf"},
        "config.h": {"file_type": "shell"}
      }
    }
  }
}
```

Opportunities

- Increased productivity & collaboration for IFS users – internal & external
  - From wherever they are, whichever device they have access to
- Seamless access for using external HPC platforms
  - EuroHPC machines via Destination Earth programme
- Ability to quickly develop efficient workflows for new modelling systems
  - ML models
  - ML training systems
  - Destination Earth Digital Twins
- Further developments planned for *web*PrepIFS
  - Ability to combine different types of experiments for complex composite experiments
    - Such as our operational production suite

Thank you for listening

- Questions?