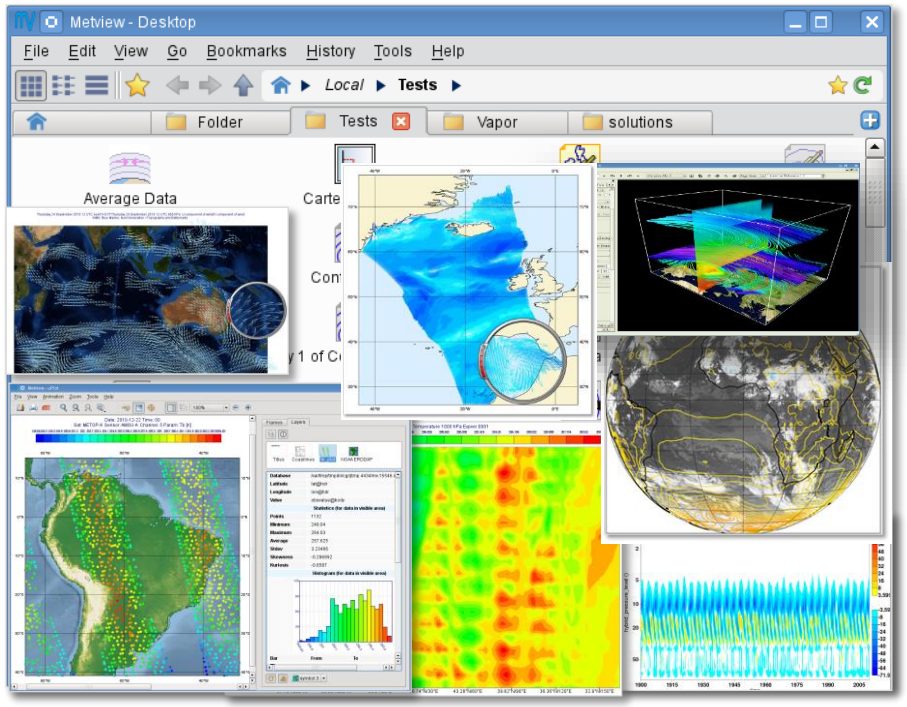
# 20 years of success for Metview's modular architecture

## ECMWF Visualisation Week

Fernando Ii

Development Section, ECMWF

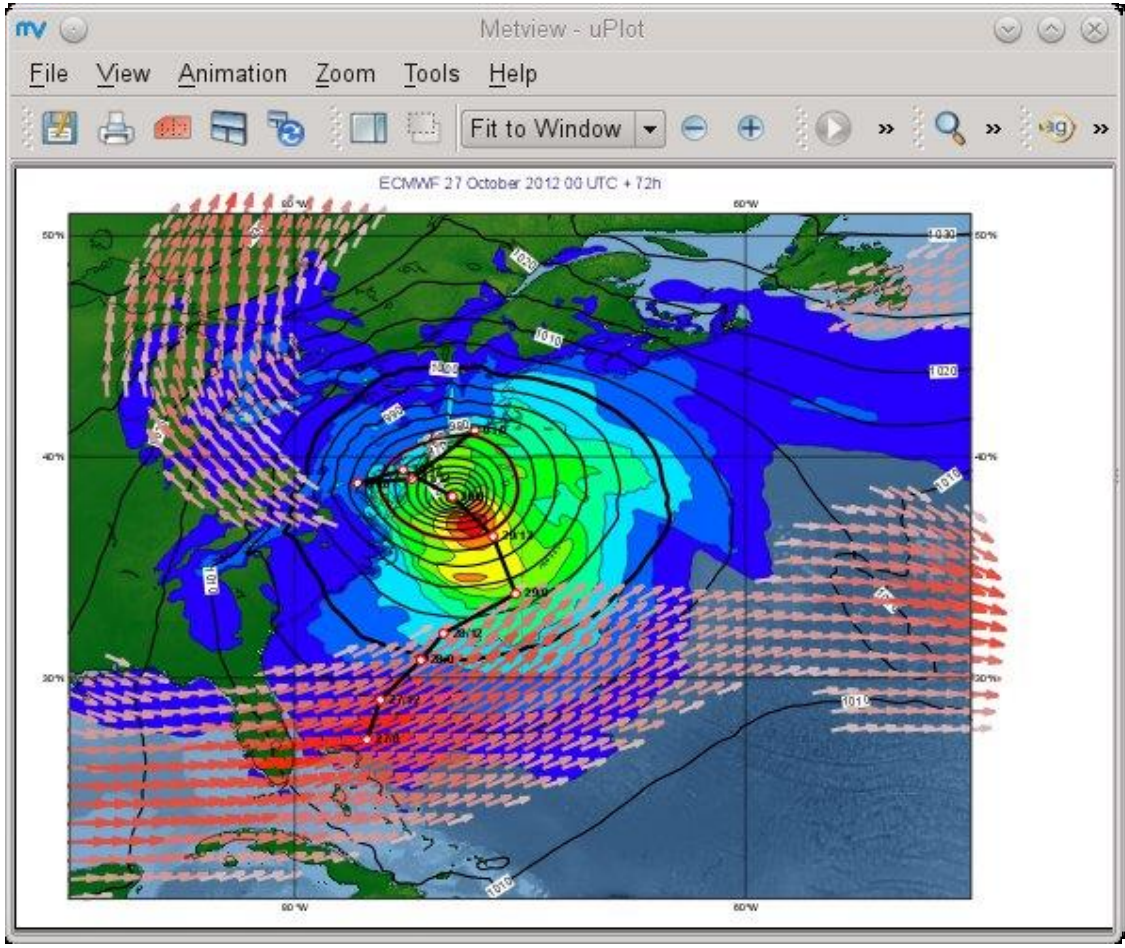


# Outline

- Background
- Conceptual Design
- System Architecture
- Summary

# Background: what is Metview?

- Meteorological data processing and plotting package
- Allows analysts and researchers to easily build products interactively and produce them in batch mode
- Can run as self-contained standalone
- Powerful meteorologically oriented language (Macro)
- Ability to overlay data from different sources and formats
- Metview is a co-operation project with INPE (Brazil)
- Open Source under Apache Licence 2.0



# Background: why was Metview developed?

- In the late 80's:
  - Most systems for visualisation and manipulation of meteorological data were based on batch jobs producing electrostatic prints
  - ECMWF-INPE MicroMagics: system to display and animate meteorological fields on an IBM-PC MS-DOS micro-computer
  - ECMWF MARS: own language to access data
  - Fast development of UNIX workstations
  - Increasing interest in development of Meteorological WorkStations (MWS) for weather monitoring and forecasting
    - fast response and user friendly interfaces
  - Forecaster's Workstation, Synergie, PROMIS, AFOSS, McIdas,…

# Background: why was Metview developed?

- Challenge:

**To develop a meteorological application combining data access, data manipulation and visualisation in a single environment based on Unix workstations**

- Announced at first EGOWS in June 1990 (Oslo)

*Metview*

*There are plans to develop a general and unique system for the visualization of meteorological data at ECMWF which should serve the scientist and the operational analyst alike. The Metview concept will provide a standard framework within which applications relating to the retrieval, processing and visualization of meteorological data can be implemented, and will enable both Operations and research*

# Conceptual design: overview

## Challenges

- Very large system
- User needs evolve over time
- Limited human resources
- Not desirable nor feasible to implement the complete system at once

## Requirements

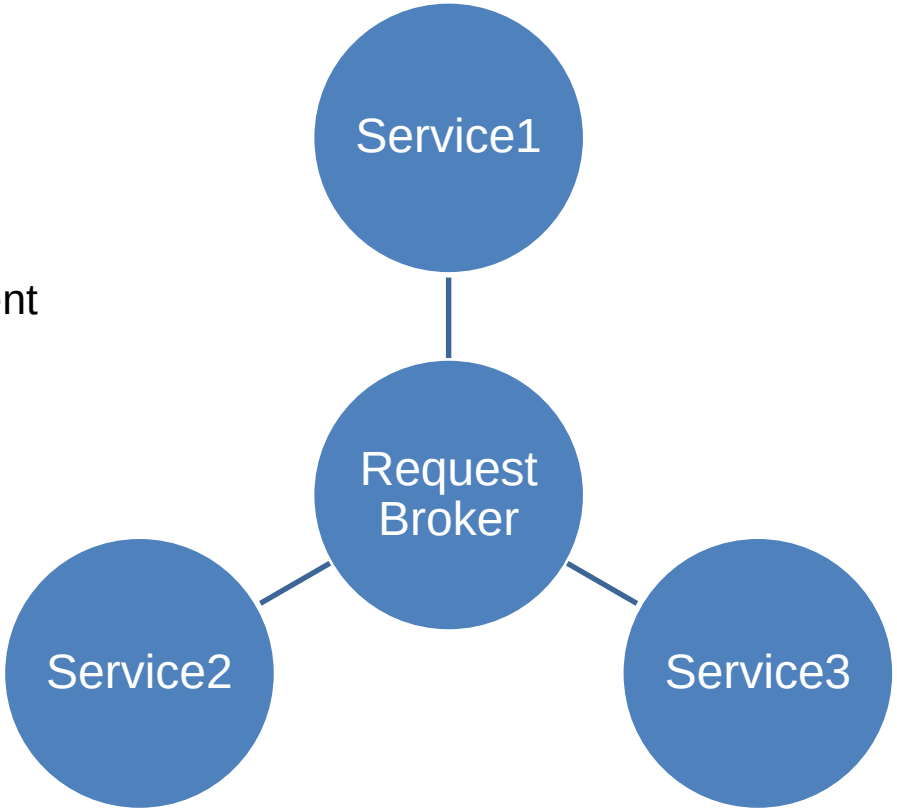
- Modular
- Extensible
- Add functionality without rewriting existing code

## Solution

- Distributed system
- Each module is a separate executable which may run on different machines
- Each module executes a service and may ask for other services to be executed by other modules
- Each module sends and receives messages

# Conceptual design: service oriented architecture

- Alternative to client-server and file-based architectures
  - Growing power of workstations (late 80's)
  - Distributing processes on a network more efficient than designing large, single-process systems
- Architecture's kernel ➡ *Request broker*
  - Provides communication between services
  - Communication between processes is asynchronous
  - Modules communicate using the TCP/IP protocol (may run on different machines)
  - User-configurable protocol
  - No "internal intelligence"



# Conceptual design: communications protocol

- Simple but powerful protocol based on a language with the following abstract syntax:

```
COMMAND,
        PARAMETER1 = VALUE,
        PARAMETER2 = VALUE1/VALUE2
        PARAMETER3 = VALUE1/TO/VALUE2/BY/VALUE3
```

- Based on MARS (Meteorological Archival and Retrieval System) language syntax

```
RETRIEVE,
        PARAM     = TEMPERATURE,
        LEVELIST = 100/850/500,
        DATE       = 20150901/TO/20150930
        TIME       = 12
```

- Used by Metview to describe requests and to implement the communication between modules

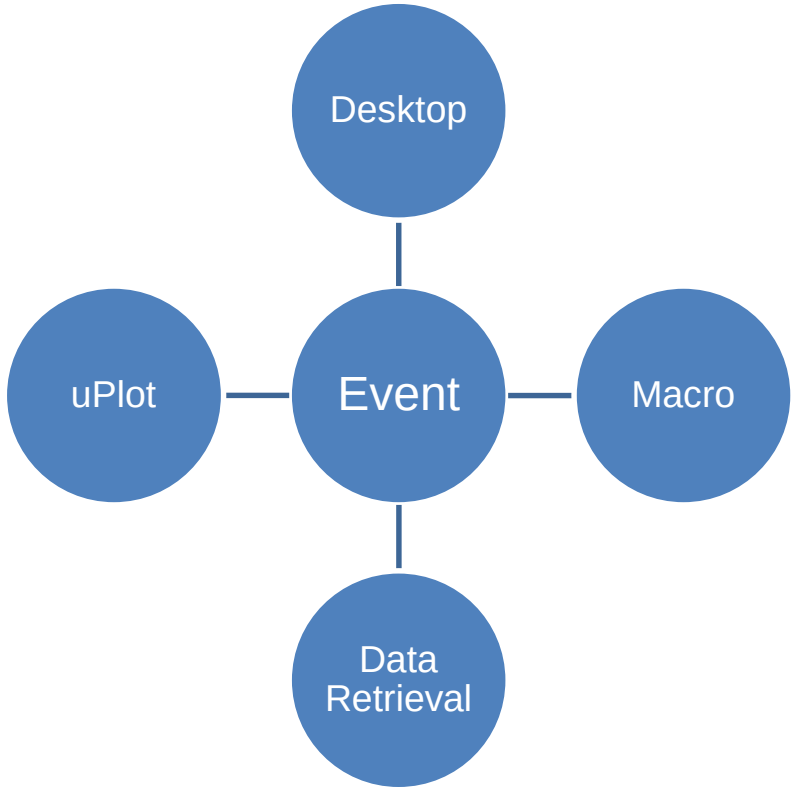
```
MXSECTION,
        DATA          = mydata,
        LINE          = 0/-180/50/110,
        TOP_LEVEL = 0.01
```

# Conceptual design: extensibility

- Fundamental feature
- New modules can be added to the system via a simple programming interface following a well-defined protocol:
  - Configuration file, which includes:
    - Registration of the new module to the request broker (e.g. name and how to run it)
    - Definition of how to build the user interface: input parameters, valid and default values, set of rules for performing a consistency check
  - “service” callback function
  - “reply” callback function
  - Meaningful icon which represents the application

# System Architecture

- Distributed system with various asynchronous processes/modules
- All processes may send and receive requests




**Event**

Assures the communication between modules
Startup: reads a *configuration file*
opens a communication port for socket connection

**Desktop**

Graphical user interface

**Data Retrieval**

Transparent access to databases/files which may be distributed on the network

**uPlot**

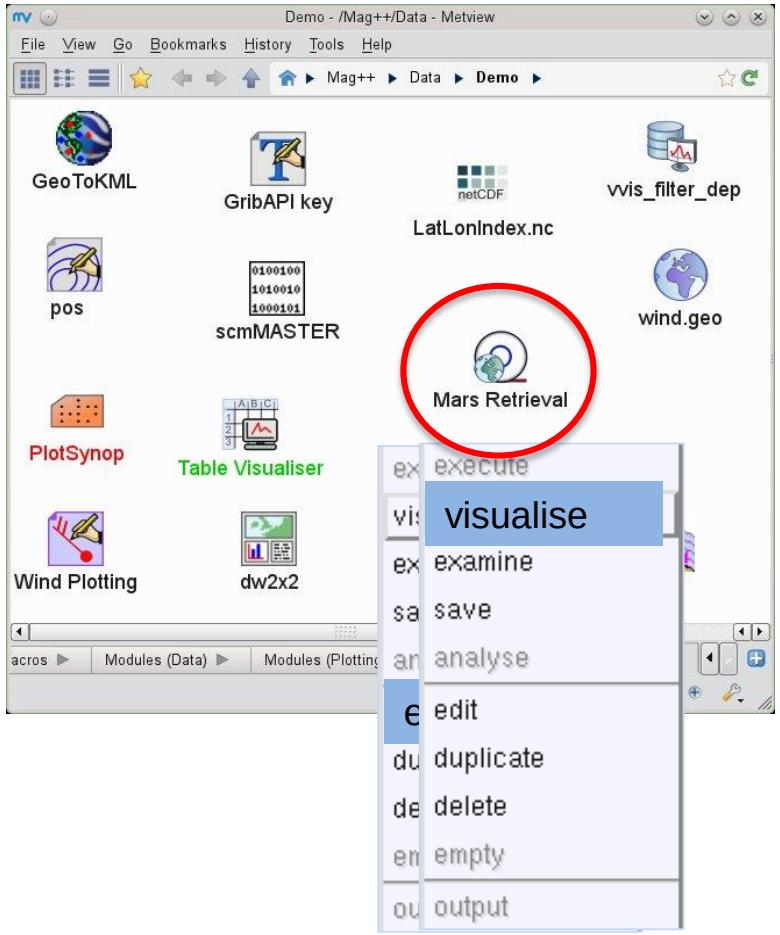
Performs all required plotting actions

**Macro**

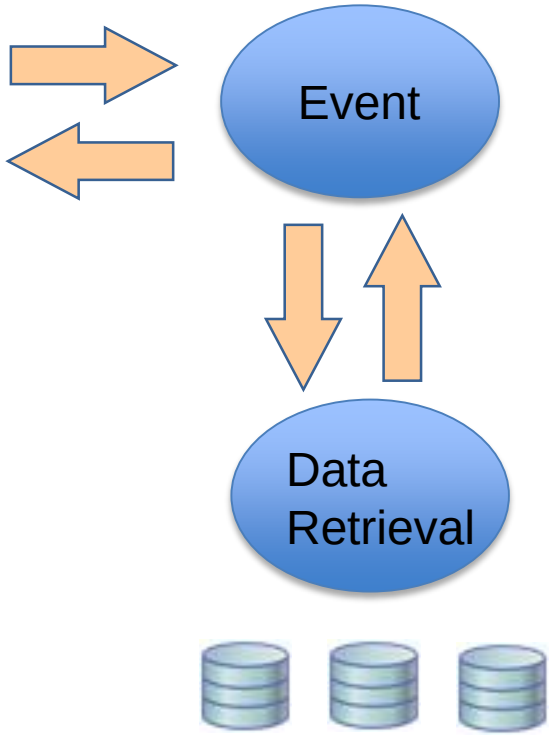
Powerful high-level meteorologically oriented script language

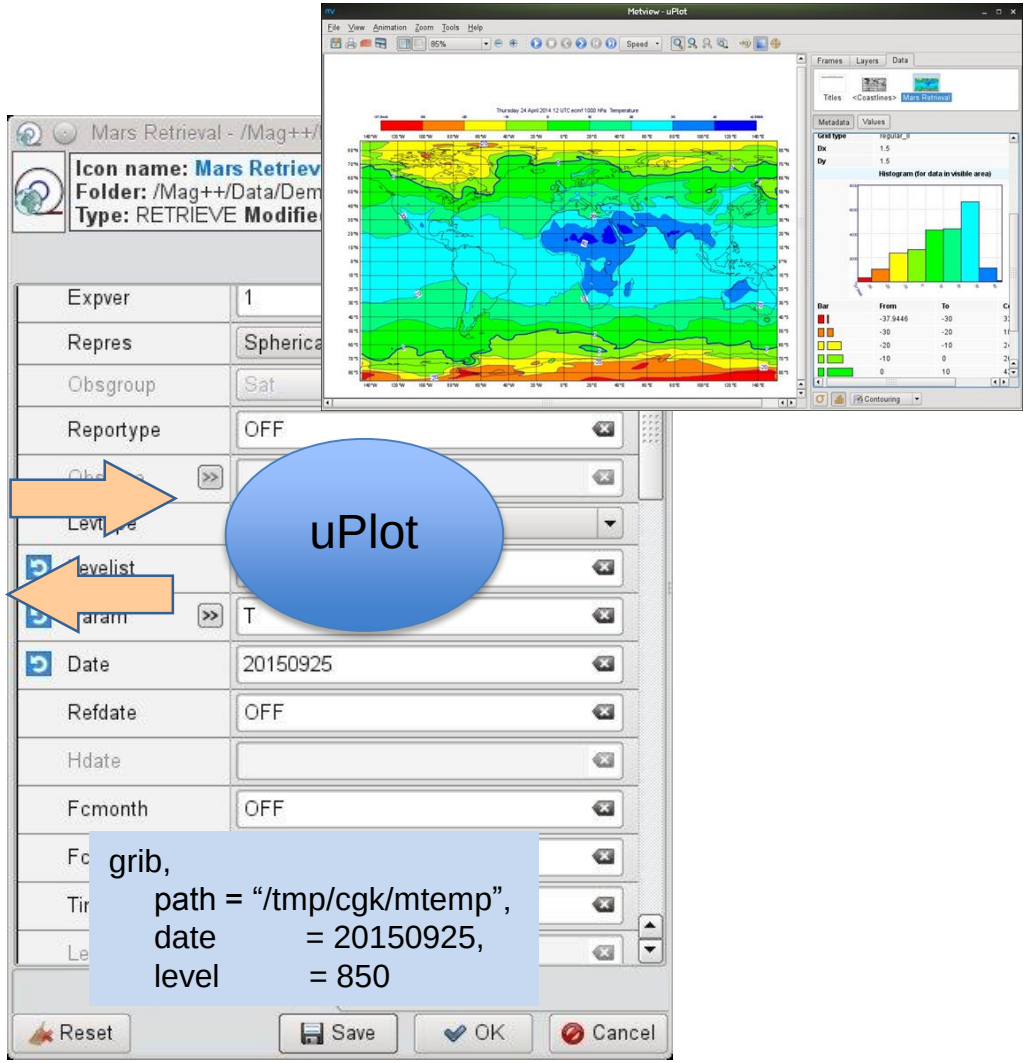
# System Architecture: Workflow

Basic Metview principles:
a) everything in Metview is an icon
b) every Metview task is a sequence of actions on icons



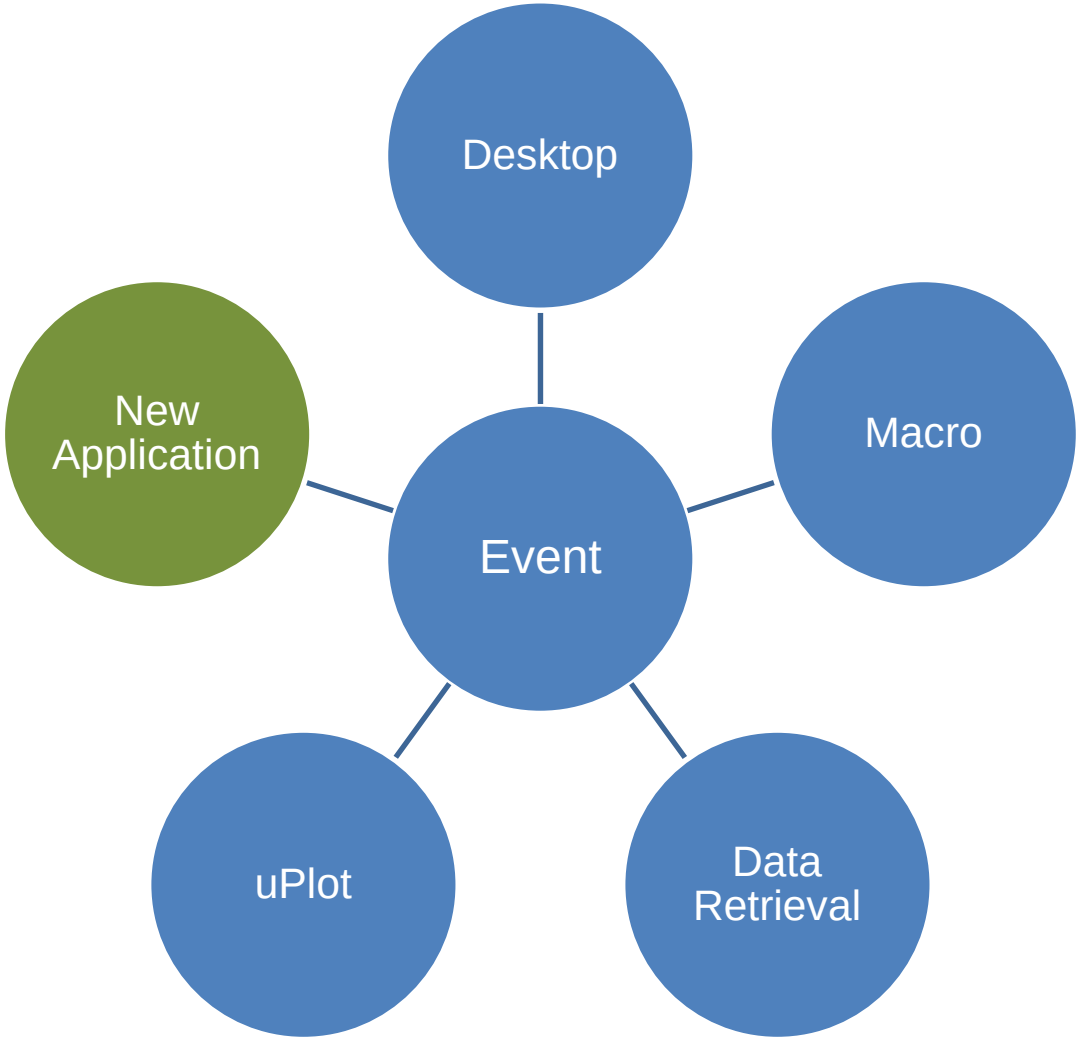

retrieve,
parameter = temperature,
date = 20150925,
level = 850




grib,
path = "/tmp/cgk/mtemp",
date = 20150925,
level = 850

# System Architecture: Easy to modify and extend



# System Architecture: Easy to modify and extend

## User Interface definition

```
GeoToKML
{
   GEOPOINTS
   [ interface = icon, …]
   { @ }

   OUTPUT_MODE
   {
      Points
      Polygon
   } = Points
   …
}
```

+

## C++ code

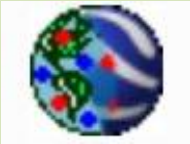
```
GeoToKML::serve( MvRequest& in, MvRequest& out )
{
   // Get input parameters
   MvRequest data = in("GEOPOINTS");
   string mode = in("OUTPUT_MODE");
   ...

   // Code to convert Geopoints to KML
   ...
}
```

+

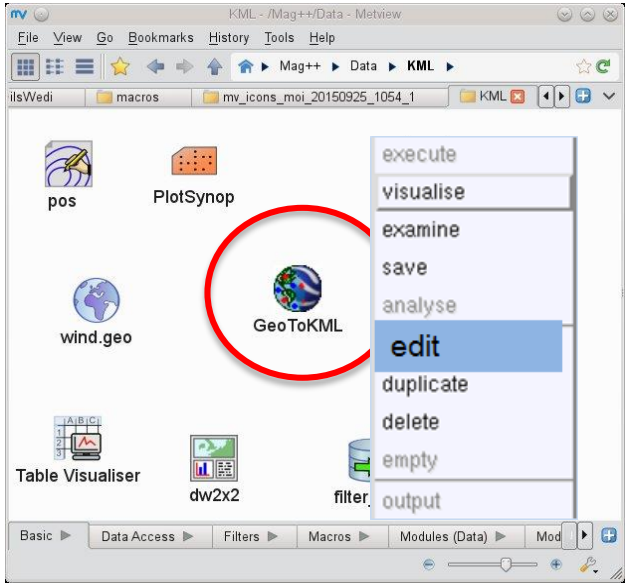
## Configuration file

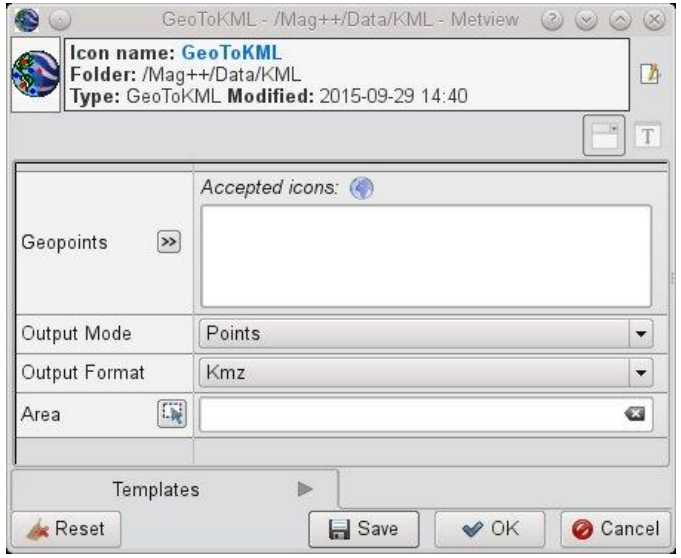
```
service,
      cmd   = '$METVIEW_CMD path/GeoToKML',
      name =  GeoToKML
```

+



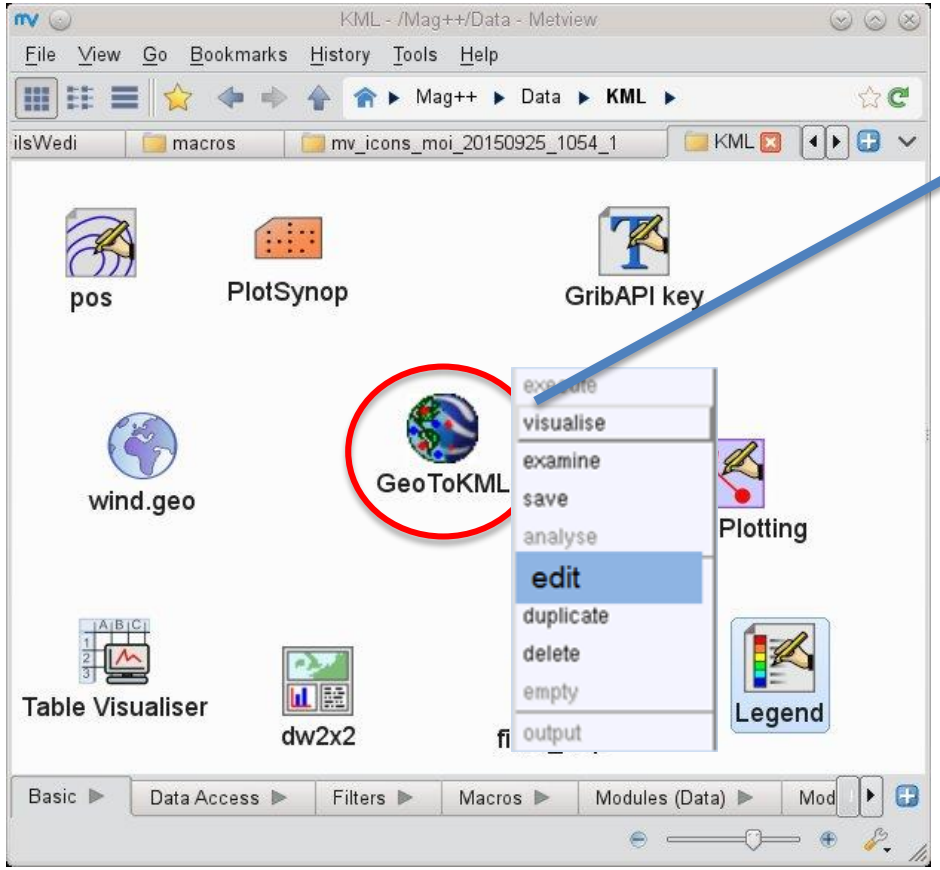
No recompilation of any existing module is required!

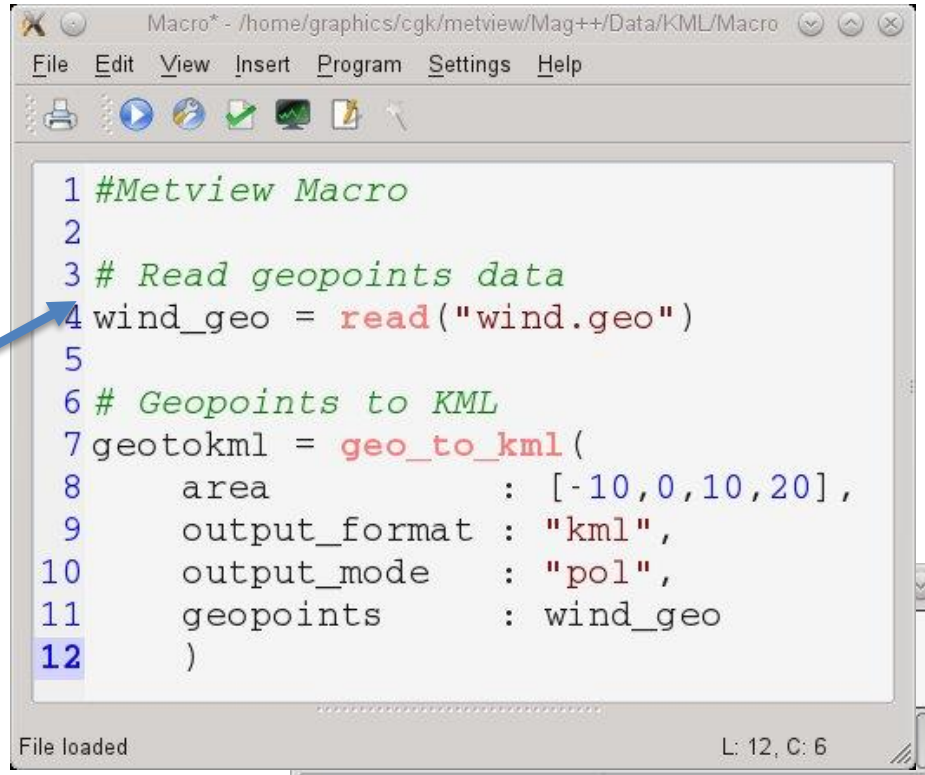


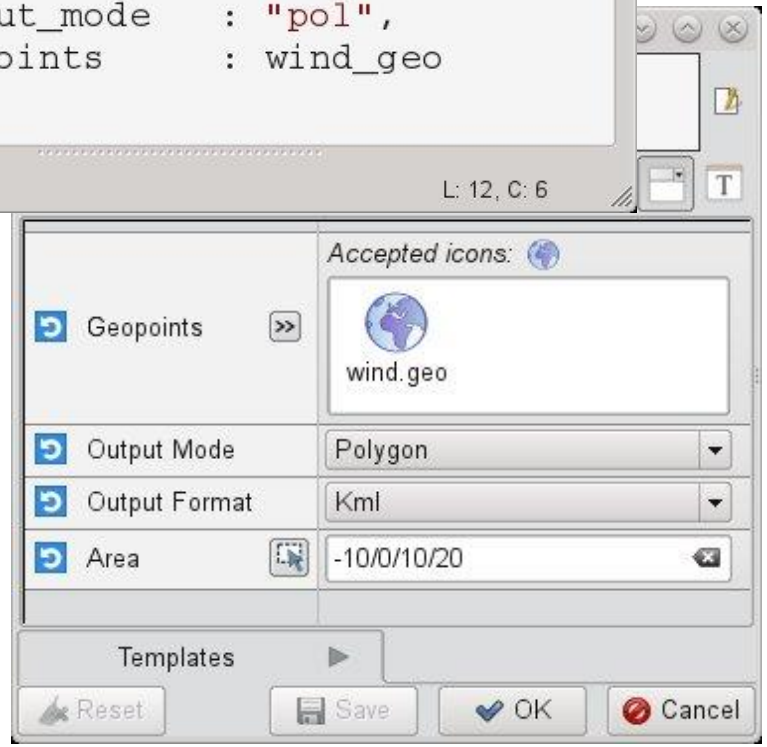



# Advantages of a modular architecture

- Strong synergy between Icons and Macros



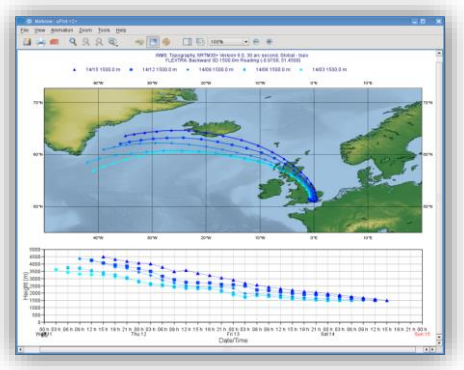



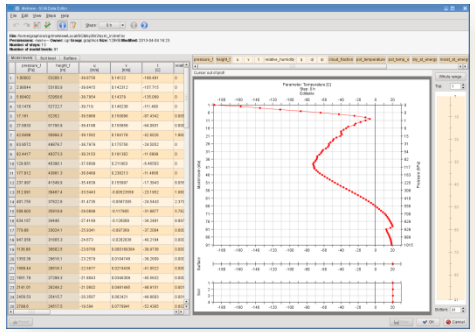
# Advantages of a modular architecture

- Interfacing external models
  - Researchers use various smaller, more specific models for their research
  - These models are seen by Metview as a ‘black box’

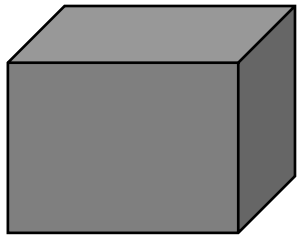
**FLEXTRA**



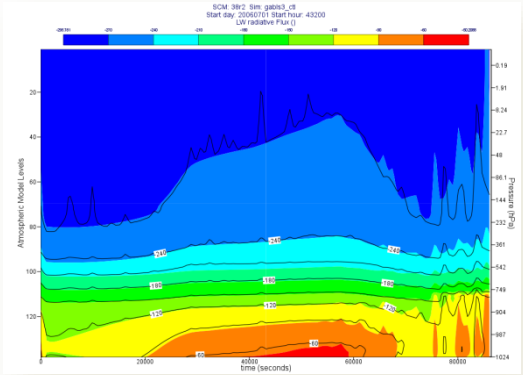
Metview prepares the input



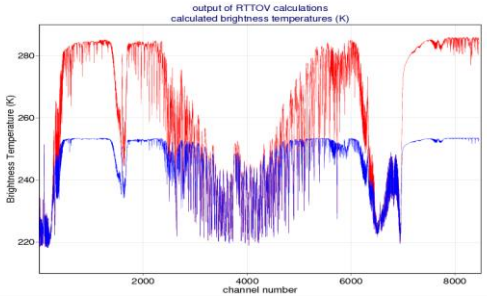
**SCM**



Metview visualises the output



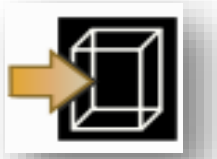

**RTTOV**



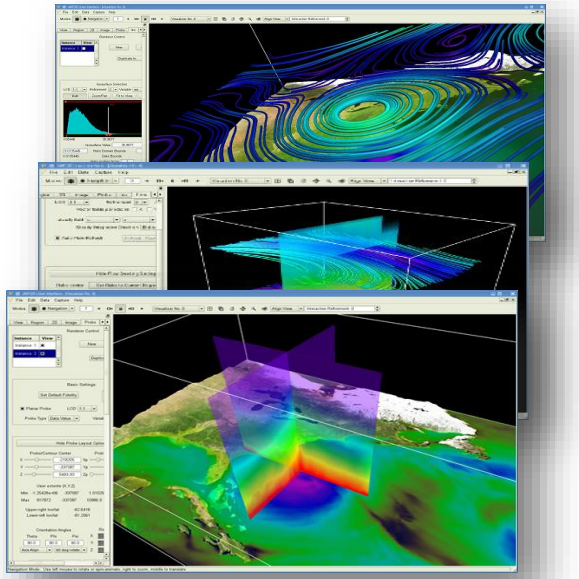
# Advantages of a modular architecture

- Visualisation using external packages
  - Vis5D
  - VAPOR 3D
  - Met.3D (future plan)
  - WMS client  built into Metview

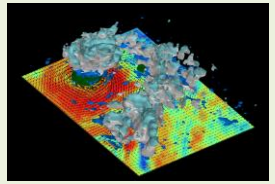
Convert GRIB data
to VAPOR format



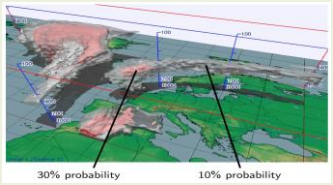
Launch VAPOR

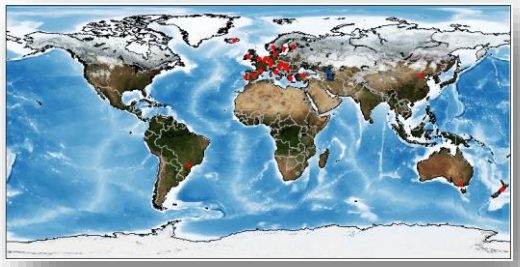


Vis5D



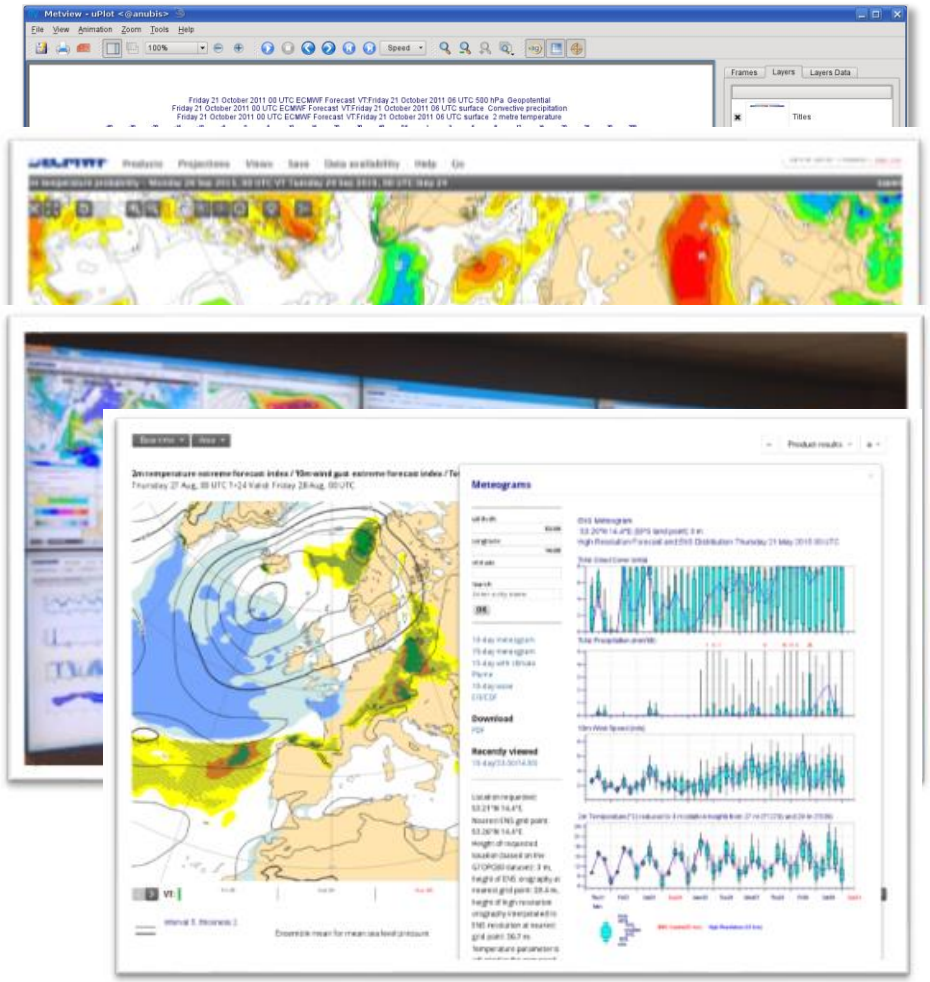
Met.3D



# Summary

- This architecture has stood the test of time:
  - MARS (1985)
  - Metview (1995)
  - ecCharts (2013)
  - WeatherRoom (2014)
  - new webplot framework (2015)
- Ongoing development of Metview to incorporate new features
- For more information …
  - Email us: metview@ecmwf.int
  - Visit our web pages: http://software.ecmwf.int/metview
  - Download (Metview source, binaries, virtual machine)
  - Documentation and tutorials available
  - Metview articles in ECMWF newsletters




Questions?