Using scripting languages for meteorological production and publication

Claude Gibert, ECMWF

Known problems

- Scientists spend a great amount of time solving problems, which are outside their domain of expertise.
- These problems are mostly related to information technology and web publication.

Known solution

- Scientists need high-level tools to support them in their technical tasks.
- Web publication should be 'automatic', in the line of current Content Management Systems.
- Details of data format and data manipulation should not require any particular knowledge from the user.

An approach

- A solution is to offer all-in-one tools, which cater for every aspect of a problem. Limiting in function but very efficient.
- The tools should offer different levels of use for different user profiles.
- The different tools should be able to work together for the whole system to make sense.
- Scripting languages have the ability to glue things together in an elegant way.

Scripting Languages

Definition

- Languages which: are weakly typed, often interpreted, often the fruit of an Open Source effort (they are free).
- Typical examples: Kornshell, TCL, Perl, Python.

Advantages

- They are portable fully featured object-oriented languages. They offer libraries implementing lots of needed components (system, network, database, file formats...).
- They facilitate rapid prototyping and iterative design. Development speed is much faster, execution time if often slower.

Gluing and wrapping

- Many tools are available (i.e. SWIG) to interface scripting languages with C, C++ and Fortran libraries like GRIB, BUFR, NetCDF, Naglib and MAGICS etc. . . .
- Scripting languages glue existing libraries and wrap them to provide an easy-to-use weakly typed interface.

Web development

- They are used in web application development (CGI, mod_perl, mod_python, Zope).
- At ECMWF Meteorological Division web applications are written using a home made framework in Perl, using object orientation.

High-level libraries

- Scripting languages offer different levels of programming:
- Fully featured object orientation for the software developers,
- Simple scripting level for the non-experts.
- Writing a script is not the same exercise as writing a program, it is more appealing because it seems 'less serious' and less involving. It is also much quicker. It encourages hit-and-miss approaches.

Small languages

- Scripting languages offer great text/string manipulation and parsing capabilities. These are priceless for web-oriented text processing (HTML, XML), but also to read configuration files or directives.
- This helps creating highly parameterised applications and flexible tools, which do not require programming.

Plot content management system

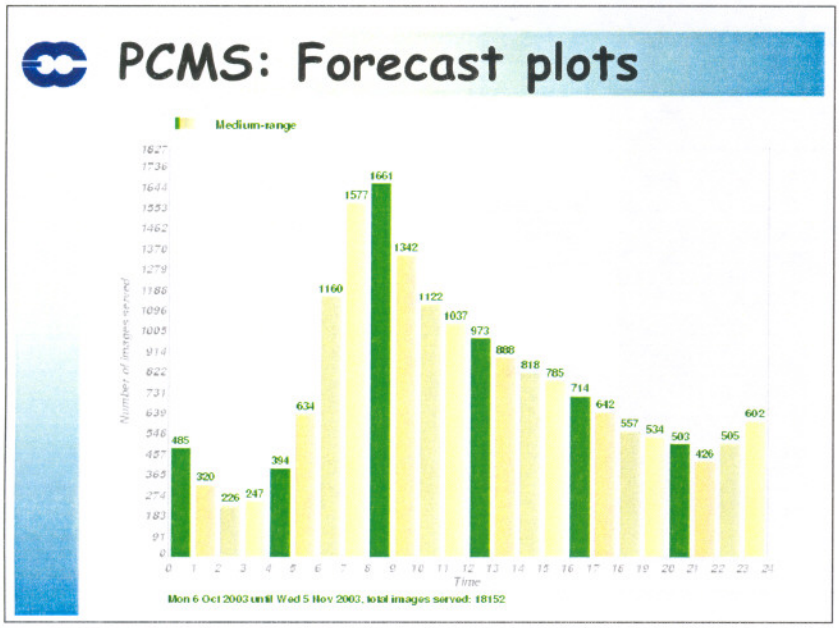
The need

- Plots are vital in Meteorology, they help monitoring and evaluating meteorological systems. They 'compress' data.
- ECMWF need to publish plots on their web site for Member States and WMO users. Internally making plots available on the intranet facilitates teamwork.
- Once a plot is on the intranet, it should be a trivial exercise to move it to the external web site.

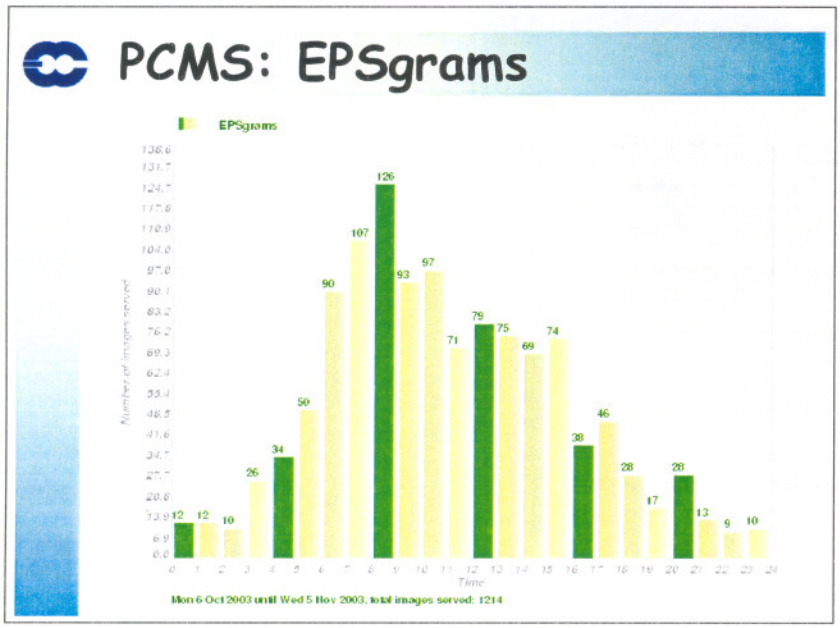
Some figures

- Our member state web site prior to January 2002 offered about 15 different products and probably 150 single images. Plot pages were edited manually.
- Our current web site hosts about 4000 products containing 230,000 single images.
- The intranet hosts 1,000 products with 18,000 images.
- We serve a peak value of 24,000 images a day. The monthly mean is about 19,000.

Forecast plots



EPSgrams



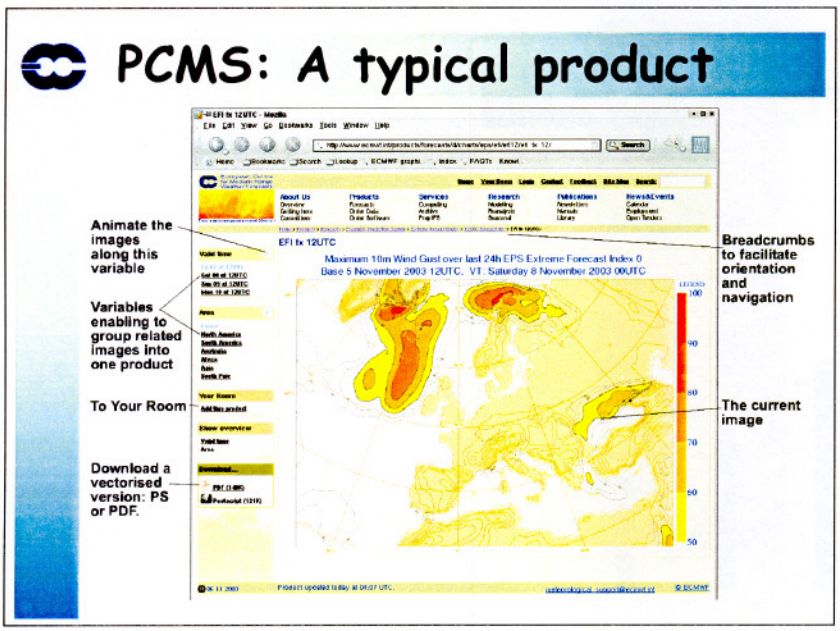
A database

- The PCMS holds:
- a database of product definitions, the meta-data, including titles, texts and references to images,
- single images, the data are held in a database of files: a file-system.
- The database is:
- populated by authoring tools inside ECMWF,
- read by a web application.

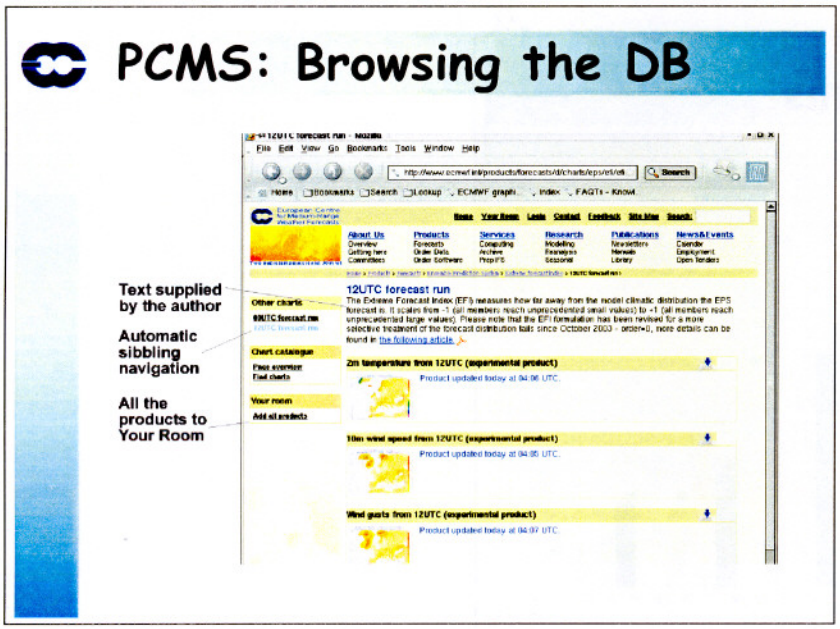
Web application

- The web application uses the meta-data (a tree of folders and products) to produce HTML pages on demand. A product represents a set of plots.
- It is written in Perl using Object Orientation. Each product is stored using object persistence in Berkeley DB.
- HTML pages are produced automatically, this helps enforcing consistency in presentation and navigation.

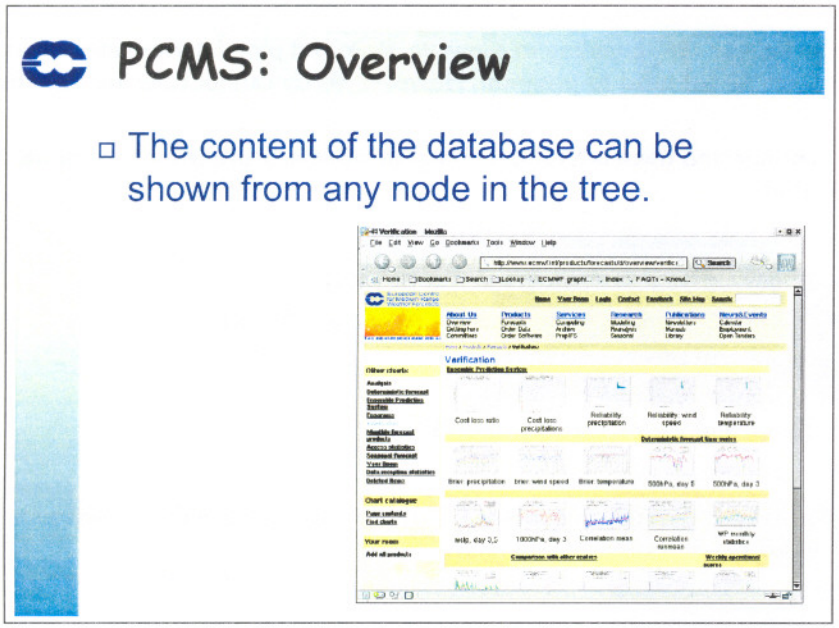
A typical product



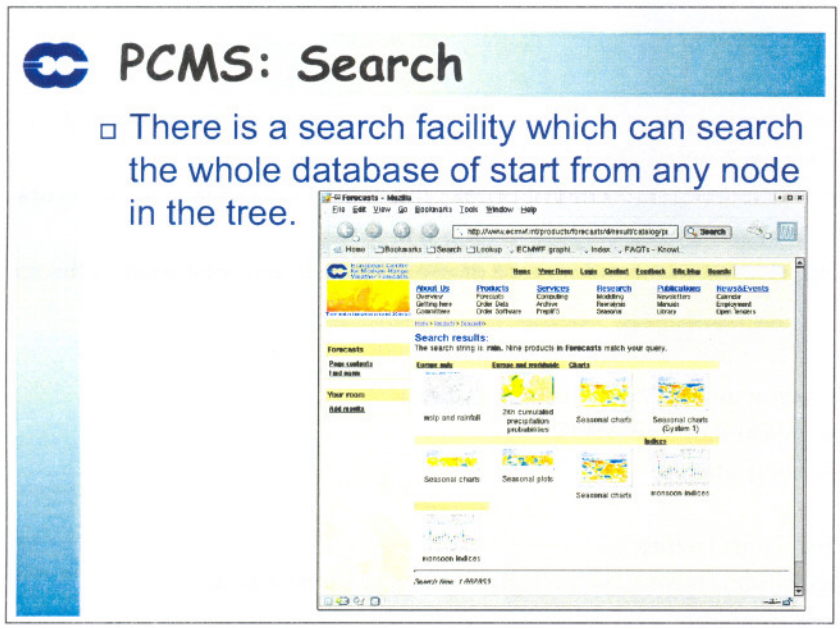
Browsing the DB



Overview



Search



Authoring

- Web authors who need to publish plots do not need any knowledge of web technologies, or image format conversion.
  - There are three levels of access:
  - programmer level,
  - access for authors who are responsible for an area of the external web-site,
  - access for intranet authors, everything is automatic. Slightly limitative.

Meta-data / data

- Data and meta-data are separate:
  - a product definition is put in the database once, and can be modified if needed.
  - images are updated on disk on a regular basis.
- Each product is associated with access control categories to define who can access it. This is because the web site is organised by contents, not by profile.

Verification

Verify: A full rewrite

- ECMWF developed a verification package in 1987 enabling to compute, store and plot deterministic scores. The package reads directive files; users do not need to program. However Verify cannot cater for specific needs.
- The package is getting difficult to maintain and there is a need to integrate other scores currently computed separately and probabilistic verification.

VeriPy: Analysis

- A verification package uses nearly all the different systems present at ECMWF:
  - retrieve and decode fields (MARS, GRIBEX) and observations (ODB, MARS, BUFR),
  - compute scores on limited areas (core activity),
  - handle a database of scores and temporary results,
  - produce different types of plots (score database access, MAGICS)

VeriPy: Useful features

- To summarise, the verification package does what most people at ECMWF need to do to either create meteorological products or run experiment diagnostics:
  - manipulate fields and observations
  - handle different databases and file formats
  - core activity
- It became clear that the building blocks for writing the verification package should be somehow, made available to end users.

VeriPy: The design

- File processing and time critical tasks could be implemented in C^{++}. VeriPy could be developed in a scripting language which would glue and wrap all the basic functions.
- Considering that typical users do not have a purely technical profile, imposing Perl and its peculiar syntax did not seem a good idea.
- Python was chosen instead for its appealing syntax and its ease of use. Python is also well used in the scientific community.

VeriPy: MetPy

- A high level library called MetPy is being developed. It currently offers:
  - an abstract definition of meteorological data and fields,
  - GRIB decoding, coding and field computation,
  - Field sorting and indexing,
  - ODB access using its NetCDF file export facility,
  - core score computing (rmse, ancf, stddev, mean), on a weighed grid and limited areas,
  - access to Verify scores (backwards compatibility) ,
  - interface with Magics and with PCMS,
  - anything available in open source.

MetPy: Early testing

- The MetPy user guide is written at the same time as the code. This enables (nice) testers to start using it before it is completed.
- This enables developer(s) to get early testing and early feedback regarding the tools offered. It is also a good way to spot forgotten features.

MetPy: An application

- Currently a typical experiment diagnostics is being written on ocean data.
  - compare ocean observation profiles with experiment fields and generate statistics,
  - It took two hours to write classes able to read the data (NetCDF on one side, GRIB on the other). These classes will be reusable for ocean diagnostics in the future.

VeriPy: Different levels

- VeriPy will offer a non-programmable interface in the same style as Verify. However, in that context, exceptional needs cannot be catered for.
- It will be possible to go from a directive level (VeriPy) to a scripting level (MetPy) and write a Python script to achieve special verification or computations (i.e. EPS probabilities).

MetPy: An example

```
from MetPy import *

an = FieldSet('an.grib'
fc = FieldSet('fc.grib')
cl = FieldSet('climatology.grib')
bi = FieldSet('biases.grib')

scores = Scores(
            fc-cl+bi,
            an-cl,area = verify_config['europe'])
print scores.correlation()
print scores.rmse()
```

Summary

- Scripting languages have changed the way developers work and the level of access of the tools offered. They are weakly typed, this enables:
  - faster development times (they require a different type of discipline from the developers),
  - 'syntactically sugared' libraries.
- They come with 'batteries included', a set of libraries for system, file, database, client-server, web access...
- The main goal is to hide technical details from people who have different expertise.
- The emphasis for developing 'good tools' should be set on:
  - offering different levels of use so that different profiles of users find what they need at different levels of involvement,
  - keeping a global vision of how tool can work together.
- The Plot Content Management System was the answer to an urgent need on the external web site. It is becoming rather popular for internal needs.
- VeriPy is still being developed. It relies on well defined building blocks made available in MetPy which keeps growing.

Future work

- Satellite data monitoring plotting system:
  - currently in Fortran with Magics,
  - plots in PCMS, difficult to maintain numbers and configuration (model).
  - MetPy links NetCDF data, Magics and the PCMS: automatic configuration.
- Offering users with unique access to all co-operative tools will be the next step:
  - adding MetPyin Metview,
  - Metview or Magics generates PCMS files,
  - offering a web-based interface.

Web site

- MetPyconstitutes a 'first' in the sense that it unifies different worlds in a scripting language:
  - fields, observation, scores and all the associated processing,
  - web related technologies: MySql, PostgreSQL, and file decoding,
  - client-server.
- This will facilitate the development of more interactive web applications.

Tool usage

- Developers use the tools they develop.
- It takes a relatively long period of time before tools are widely used.
- for the PCMS, the system had been in place for about a year when users started to use it for the intranet (there was not strictly speaking a need),
- we expect that it will probably take as long for MetPy(although there is a need).

PCMS: example

```
product:
  parent       =        eps/efi/efi12
  name         =        efi_fx_12
  long_title   =        Wind gusts from 12 UTC
title          =        EFI fx 12UTC
content        =       step(48,72,96,120)
    area(Europe,North America,South
    America,Australia,Africa,Asia,SouthPole)
    param(fx)
    time(12)
    suite(pop)
    task(efi_fx)
access         =       real_time
input:
  path         =       images/efi/12/output.1.ps
  content      =          param(fx) area(Africa)
  step(48,72,96,120)
  scale        =       95
  rotate       =       90
```

Adding tools

- With the PCMS tools available it became trivial to write a small Visual Basic macro running in PowerPoint to export a presentation to the VMR format.
- Object orientation enabled to subclass a standard product to display it like a presentation.

An application

The product is presented in terms of a presentation. The overview becomes the 'slide sorter' and the arrows (right) enable one to move to the next slide.