METVIEW - Plot Module
Design Document
Version 1.3 - June, 1997

Purpose
The purpose of this document is to describe a new module for visualisation and plotting meteorological data in Metview. The document contains a functional description (including the user interface), a description of the interfaces between this module and the other METVIEW modules, and a design strategy for the internal module structure.

Caveat
This is a working document. According to modern programming practices, software development should be done in an evolutionary way: the system should grow from an initial design and implementation through repeated redesign and implementation. Therefore, some parts of this document may be incomplete at any given moment, and the document will be reviewed as the implementation evolves.

Table of Contents
1. Introduction
2. User's Perspective of Current Version
3. System Design
   3.1 Conceptual Design
   3.2 Colour Choice
   3.3 User Interface
4. System Interfaces
   4.1 Description of METVIEW Requests Processed by PlotMod
4.2 Detailed Examples of METVIEW Requests

4.3 Relation PlotMod - MagProc

5. Software Implementation

6. Conclusions, Acknowledgments and References

Appendix - C++ Class Interfaces

Last Update 26 June, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts (ECMWF), Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
METVIEW - Plot Module

1. Introduction

The development of a new visualisation and plotting module for METVIEW, which is upwards compatible with the current VisMod, is a logical consequence of a software project's normal development curve. The Software Engineering literature contains many reports that document that an interactive program can only be completely specified when a working version is running.

Therefore, the standard recommendation for interactive program development (see, for example, Brooks [1992]) is that an interactive system should be re-written after a working version is submitted to substantial user trials, which provide an assessment of its usability and adequacy. The working version is therefore considered as a basis, in terms of functionality and interface, for a second version, which is then designed to account for performance, maintenance and reliability.

The current VisMod was developed, in its major parts, on the period 1992-1995, whilst the METVIEW design was evolving. During the development period, a number of additions were made to the software, which had not been foreseen in the original design. These changes inevitably affected the software's structure and have made it necessary for a second version to be developed.

This document contains the design for this second version of the visualisation and plotting module in METVIEW, based on the following assumptions:

- The interface and functions should be preserved (and revised, when needed), based mostly on an usability analysis of the current version.
- The interface to the GenApp module, based on requests and the drag-and-drop widgets, should be maintained.
- The interface to the MagProc module should be revised.
- The internal structure should be completely reviewed. Although reuse of X-Window related widgets will be attempted, a new design is needed.
- The resulting software should be portable across a wide range of UNIX-based operating systems (IRIX, DEC/OSF, HPUX, Solaris, AIX and Linux), should be easy to be maintained.
- Given user expectation of typical interactive software, efficiency considerations will play an important role in the PlotMod system design.

The new module will provisionally be called PlotMod. The idea of developing the module under a different name stems from Baudoin Raoul, and allows the current VisMod and the new PlotMod to be available concurrently, and thus make the transition a smoother one.

The document structure is as follows:

- Chapter 2 provides an user perspective of the functionality of the current VisMod, based mainly on the users perspective, but also on the experience of the Graphics Group at ECMWF and CPTEC.
- Chapter 3 presents a proposed object-oriented design strategy for the new module. This design has taken into account, whenever feasible and application, the new developments in the Software Engineering area, especially those related to the emerging discipline of software patterns.
- Chapter 4 contains a description of the relation of PlotMod to the other METVIEW modules, with a view of maintaining compatibility with the current METVIEW implementation, especially in the processing of requests and macros, the relation to the GenApp module ("drag-and-drop"), and the relation between PlotMod and the MagProc module.
- Chapter 5 describes the software implementation components of PlotMod.
- Chapter 6 contains some concluding comments and acknowledgments.
- An Appendix contains the headers of the C++ classes used in the development of PlotMod.

Last Update 21 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts (ECMWF), Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
2. User's Perspective of Current Version

2.1 Introduction

A limited usability test was conducted at ECMWF to assess the current uses of METVIEW, especially in what relates to visualisation and plotting issues. Each user was asked to show how he uses METVIEW in a typical working session, explain what problems he is currently facing, indicate possible improvements and corrections he needs or requires.

On average, each interview lasted one hour. Inevitably, the users expressed opinions over the complete METVIEW software. For the purpose of this document, this section concentrates on the topics involving the current VisMod.

The users interviewed were:

- Research Department, Physical Aspects Sections: Christian Jacob, Pedro Viterbo.
- Research Department, Data Assimilation Section: Erik Andersson.
- Research Department, Satellite Section: Elisabeth Gerard.

This sections also draws on the experience of CPTEC users, including:

- Research, Model Division: Jose Bonatti.

This section presents the users considerations, grouped by topics. In what follows, software similar to METVIEW, such as Vis5D, GRADS and McIdas, will be mentioned, whenever it is relevant to compare their functionality with that of METVIEW. These mentions are not intended to provide a basis for claims of superiority of one software over the other, but rather as a result of user experiences and opinions, especially when the user has had access to different systems.

2.2 General METVIEW Use

Overall, there seems to be some significant trends in METVIEW usage:

- A general satisfaction with the user interface, seen as "user-friendly" and having a small learning curve.
- The ever-growing use of the macro language as a substitute for FORTRAN programs for manipulation and post-processing of model data.
- An emphasis on shaded contour maps as the most common method of generating pictures from model data.
- A requirement that METVIEW should provide as close to a WYSIWYG ("what you see is what you get") performance as possible, achieving a correspondence between screen views and plotter output.
• An overall concern about graphical performance, especially in terms of the time it takes to draw (or redraw) 2D maps.
• A requirement to improve the facilities for displaying and manipulating satellite imagery.
• A requirement to allow user more control of how applications display data, including the possibility of displaying different perspectives from the same data set.

The ubiquitous topic of the "paper-less office" was mentioned on most discussions with users. As a general view, it was felt that the workstation graphical capabilities gave the user a chance of experimenting various different visual definition, before selecting those that were fit to print. There were users (especially on Meteorological Operations) in which case the quantity of screen-generated output greatly exceeded the number of plots produced.

Therefore, a typical METVIEW user cycle could be expressed as:

• Preparation of a macro program
• Running a macro
• Visualization of the output
• Plot preview of some fields
• Plotting chosen maps.

Specific comments for each issue are presented below. In what follows, the main emphasis is placed on the issues related to the current VisMod, although it is inevitable that some issues are related to other METVIEW modules.

### 2.3 User Requirements for a New VisMod

The usability test concentrated on questions regarding the users satisfaction and wishes as regards the VisMod module, which are outlined below.

#### 2.3.1 Plot Window Layout and User Control

In general, it was felt that VisMod does not allow the user sufficient control over the output displayed. The interaction of VisMod with applications other than standard 2D contouring (and area fill) is not satisfactory; dropping application icons into a plot window may generate undesirable results. Therefore, currently most users use separate plot windows for displaying different application results. The unpredictability of VisMod behaviour is a very negative aspect and the new module will need to have a well-established behaviour pattern.

#### 2.3.2 Graphics Performance

It has been acknowledged many times by all METVIEW and MAGICS users that the graphical quality of output is extremely good, and this has been a major positive aspect of METVIEW. However, there is concern from users on the graphics performance of METVIEW, especially as regards interactive screen plots.

Compared with similar packages (such as GRADS), METVIEW is one order of magnitude slower when drawing a shaded contour plot. In one experiment performed at CPTEC, with data already on the local disk, the same shaded contour took 3 seconds on GRADS and 15 seconds on METVIEW.

The issues involved here are not simple and a straightforward answer to this problem is not possible. One of the problems is that when producing screen output, interactive users prefer speed to quality. The inverse rule applies to generation of paper output. As a general guideline, it was felt that the new PlotMod should allow for a compromise between screen output and paper output, allowing different graphics engines to be associated with it.
2.3.4 Animation

Animation in METVIEW is used not so much as a means of sequence visualisation but as a way of rapidly browsing through a set of files. In this respect, there is an overlap between the functionality provided by the slider in normal operation mode for a plot window (where a user would browse all his graphics), and the animation is frame-by-frame mode.

Overall, it was felt that the animation could be merged with the visualisation, since it not felt to provide a significant benefit from being a separate function.

2.3.5 Imaging

The image manipulation functions available on the current VisMod fall short of the typical ones necessary for most users which work with images. Whilst a full-blown image processing module for METVIEW would require a significant development, the gap between what is available and what most users would regard as sufficient is not so great, and includes:

- Provision for image manipulation functions in macros.
- Interactive interfaces for defining enhancement curves.
- Support for colour imaging in METVIEW, as colour composites and false-colour enhancement.
- Facilities to combine NWP fields (e.g., cloud cover) with satellite images.

2.3.6 Colour Manipulation and Presentation

Most users felt that the colour coding in METVIEW did not satisfy their requirements. METVIEW should, in their view, provide a continuous-tone colour palette from which a user could choose the desired set of colours (typically by providing a start and an end colour). The legend should also follow the same convention and be presented as a continuous-tone legend, rather than a set of colour boxes.

Typical systems which provide continuous colour choice and legend to the user are Grads and Vis5D. The fact that these systems do not always allow complete flexibility as METVIEW does (e.g., for defining the direction in which the colour wheel is traversed) is less important, in the user's perspective, than a wider choice of colours.

2.3.7 Compatibility between batch and interactive processing

There are currently two different ways in which graphical output is produced in METVIEW currently, one by defining a PlotWindow and the other by defining a SuperPage. It is a strong wish from the users that these commands are unified and that the current differences in batch and interactive processing removed.

2.3.8 3D Visualisation

3D visualisation is coming of age in operational and research meteorology and many users are now starting to rely on it as a means of obtaining important information for large data sets. ECMWF has started to use 3D on a daily basis, especially from examining ensemble forecasts and propagation of forecast errors. The most widely used 3D visualisation packages on the meteorological community is Vis5D. Given that it already provides a substantial set of functions and that it is freely available, it was felt that an integration between Vis5D and METVIEW is highly desirable.
2.3.9 XY Plots

Many users felt that the current XY plots provided by METVIEW need to be improved. Two problems were especially outlined: the limited interface tools available for defining the axis composition and the curve displays, and the lack of flexibility for combining different information.

The combination of METVIEW with third-party packages (such as PV-Wave and Xmgr) was proposed as a means of improving its XY plotting facilities.

2.3.10 Macro Generation from PlotWindow

Some users stated that they felt that the facility of generating a macro from a plot window did not produce satisfactory results, as the macros produced were considered too complex for the majority of users. They suggested that this facility should provide simpler code, which the user could more easily modify.

2.4 Resume of Users Perspective

The following list indicates the most significant improvements and additions to VisMod which were requested by the consensus of METVIEW users. They are given on a priority list, which is based on two criteria: the users' demand and an expectation of what is realistic to expected, given the development constraints:

- Improvement on the graphics performance.
- More flexibility for displaying results from applications (especially for non standard 2D maps).
- Better support for X-Y plots and ASCII data.
- Improved colour facilities.
- Consistency between batch and interactive processing.
- Consistency between normal visualisation and animation.
- Provision for inclusion of 3D visualisation software (Vis5D).
- Better support for text and legend display, including more informative texts and continuous-tone legend support.
- Additional image processing functions.

Last Update 25 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
METVIEW - Plot Module

3- System Design

3.1 Conceptual Design

3.2 Colour Choice

3.3 User Interface

Last Update 14 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
METVIEW - Plot Module

3.1 Conceptual Design

3.1.1 Introduction

3.1.2 Pages, Subpages and Superpages

3.1.3 Pages and Views

3.1.4 Views and Applications

3.1.5 Page Hierarchy

3.1.6 Matching Rules and Subpage Creation

3.1.7 Layout Definition

Last Update 21 March, 1997 by Gilberto Cámara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
3.1 Conceptual Design

3.1.1 Introduction

The conceptual design for PlotMod is best understood by considering a simple example: suppose a user wants to display the same DataUnit (model analysis, temperature variable, 10 pressure levels) in three different perspectives: a 2D contour plot for a selected geographical area, a cross section and a vertical profile. He wants all output plotting areas to be related, so that the same layout could be applied to different data sets.

The user would also like an output layout similar to the one shown in Figure 3.1, where the topmost part of the plot window shows a vertical profile and a cross-section, and the lowermost part contains three of the possible 10 pressure levels. He would use a scrollbar (not shown on the figure) the display all 10 different pressure levels, as he does in the current VisMod.

This plot window organization, although simple to conceive, is impossible to be obtained in the current VisMod. In order to cater for this kind of layout, PlotMod will rely on some concepts:

- Pages: independent parts of the drawing area, where data is shown (the above figure has three pages, two in the top part and one in the bottom).
- Subpages: data-dependent drawing areas, which are the places where a page places its graphical output (in Figure 3.1, each of the top pages has one subpage, and the bottom area has 10 subpages, out of which 3 are currently visible).
- Views: each METVIEW application is associated to one view, which defines how the graphical output is produced. Each page is also associated to a view. In Figure 3.1, there are three views: the upper right page is mapped to an XZView (associated with the cross section applications), the upper left page, to a PZView (associated with applications such as Vertical Profile), and the lower page to a XYView (used for applications which display onto a XY geographical area, such as field contouring, image and observation plotting and colour wind).

These concepts are described in more detail in the next sections.
Figure 3.1 - Example of one data unit, different views

Last Update 21 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
3.1 Conceptual Design

3.1.2 Pages, Subpages and Superpages

PlotMod displays meteorological graphics in *plot windows*. Each plot window will have its own user interface, and will be completely unrelated to other plot windows. In order to allow for different DataUnits to be displayed in one plot window and for the same DataUnit to be displayed in different views, PlotMod uses the concept of "pages". A *page* is the entity which can receive requests to display DataUnits (and associated VisDefs).

The example shown in the lower part of Figure 3.1 shows a case where a DataUnit contains fields for different pressure levels which are to be plotted separately, within the area allocated to a page. In order to support this behaviour, we need to introduce the concept of *subpages*. A *subpage* will contain a single plot of a meteorological variable (or a matching combination of meteorological variables). Subpages are data-dependent and are created by their parent pages according to matching rules (described below). Each subpage is associated to one or more drawing areas (or canvas), and has associated text and legend areas.

In graphical terms, a page corresponds to an area of the display surface (screen or paper), and a subpage will be a window which is contained completely inside it. From the point of view of the window manager (Motif), each subpage will be a separate drawable (X window or X pixmap), and will respond independently to events such as Expose and Resize.

Note that, whilst the actual number of subpages is data-dependent, not all subpages of a given page are displayed simultaneously. In a similar fashion to the current VisMod, the user will be able to decide how many subpages will be visible. A scrolling bar (similar to the current animation bar in VisMod) will allow the user to display all subpages. Details of establishing the visual layout are discussed in section 3.1.7.
3.1 Conceptual Design

3.1.3 Pages and Views

One very important problem in METVIEW is the control of the visualisation output. In a way similar to desktop spreadsheet applications, each page in PlotMod is a associated to a View. A view is a particular way of presenting a meteorological variable, and each application is associated to a View. Table 3.1 below describes the views supported in METVIEW and the applications associated with it.

By default, when a page is created, it is not associated to any view. The user will be responsible for defining which view is associated to each page, and for establishing its parameters (e.g. area and projection in the case of XYView, scaling of axis in the case of Axis View). This behaviour contrasts with the current ViewMod, where by default a plot window is created with an XYView (with a default area of the whole globe in cylindrical projection).

TABLE 3.1

<p>| VIEWS AND APPLICATIONS IN METVIEW |</p>
<table>
<thead>
<tr>
<th>VIEW</th>
<th>Description</th>
<th>Parameters</th>
<th>Related Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrossSectionView</td>
<td>A cross section of the atmosphere, based on a straight line on lat/long coordinates.</td>
<td>1. Line on earth's surface (in lat-long coordinates) 2. Initial and final levels 3. Vertical axis organization (linear/log)</td>
<td>Cross Section, Averages</td>
</tr>
<tr>
<td>TephView</td>
<td>Two drawing areas with the special coordinate system used by the tephigram application</td>
<td>1. Minimum and maximum temperatures 2. Top and bottom pressures</td>
<td>Tephigram</td>
</tr>
<tr>
<td>VerticalProfileView</td>
<td>An axis where the horizontal coordinate is a parameter and the vertical coordinate is a height</td>
<td>1. Top and bottom pressures 2. Vertical axis organization (linear/log) 3. Point on earth's surface (lat/long)</td>
<td>Vertical Profile</td>
</tr>
<tr>
<td>TimeAxisView</td>
<td>An axis where the horizontal coordinate is a time measure, and the vertical coordinate is a parameter value</td>
<td>1. Vertical P Axis (scale, min and max values) 2. Horizontal T axis (initial and final time, increments) 3. Graph type (curve, bar chart, area). 4. Graph parameters: colour, thickness, line type and smoothing, symbol</td>
<td>Metgram</td>
</tr>
<tr>
<td>AxisView</td>
<td>A general axis where the user may establish what to use for horizontal and vertical coordinates</td>
<td>1. Horizontal and Vertical Axis (scale, min and max values, increments, colour, symbols) 2. Graph type (curve, bar chart, area). 3. Graph parameters: colour, thickness, line type and smoothing, symbol</td>
<td>Curve</td>
</tr>
</tbody>
</table>

Last Update 21 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for
Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
METVIEW - Plot Module

3.1 Conceptual Design

3.1.4 Views and Applications

General Description

In this design, the applications will be separated from the views. After the user has associated a view with a page, this page will now be able to accept Data Unit drops and Application drops. Applications should only provide data, which will be displayed on a plot window which has a compatible view. Therefore, there are two possible ways of providing data to PlotMod:

- Dropping DataUnits into a plot window
- Dropping Applications into a plot window.

Case 1 - Dropping DataUnits into a Plot Window

When the user drops a data unit icon into the page, there are some possibilities:

- The page has no associated view. In this case, the page will be assigned to the default view (XY View), and the default application (Field Contouring) will be called to provide the necessary data.
- The page has a view, but no data. In this case, the matching rules apply, and the default application for that view is called to provide the necessary data.
- The page has a view, and data (associated with an application). In this case, the matching rules apply and the current application associated to the page will be called.
- The page already has an associated view which is incompatible with the data unit. The drop is then rejected.

Therefore, to obtain a normal field contouring, the procedure required would be:

- Define a XYView associated to a page (geographical area and projection).
- Drop a DataUnit and associated VisDef into the page. The contouring application will be called to produce the desired output.

Case 2 - Dropping Applications into a Plot Window

When the user drops an application icon into the page, there are three possibilities:

- The page has no associated view. In this case, the page will be assigned to the default view associated with the application, and the application will be called to provide the necessary data.
- The page already has an associated view which is compatible with the application. In this case, the matching rules apply, and the application is called to provide the necessary data.
The page already has an associated view which is incompatible with the application. The drop is then rejected.

The procedure for producing a cross-section would be:

- Define a XZView on a page of a plot window. This would be done by an interactive interface where the user chooses the geographical line in lat/long, the scaling associated to the Z axis.
- Create a new Cross-Section application in GenApp.
- Associate a DataUnit to this cross-section application.
- Drop the cross-section icon into the page. The cross-section application will be called, and the XZView parameters passed to it by PlotMod to produce the correct output.

Additional Notes on Applications and Views

The possibility of dropping both DataUnit and Application icons to one single page creates an asymmetric situation, which may be the case of later problems. The alternative would be to create a new icon for the default application (Field Contouring), and to allow only applications (and no Data Units) to be dropped into a plot window. However, this could create a behaviour which is very different from the current VisMod, and might confuse current users.

Last Update 21 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
3.1 Conceptual Design

3.1.5 Page Hierarchy

In figure 3.1, the same data is displayed in three different pages. Although this could be achieved by means of the separate drops on the same DataUnit into each of the pages, PlotMod will provide a way to obtain the same result with only one drop. To that end, the plot window is organized as a page hierarchy, illustrated in Figure 3.2.

Figure 3.2 - Page Hierarchy for PlotMod.

The top-level node of the hierarchy is an abstract level, referred to as Root. The objective of this top-level node is to provide default descriptions for all elements of the page hierarchy (such as supplying the default view of a page, or the default coastlines for a subpage).

The next level on the page hierarchy is the instance of the plot window itself. This instance is referred to as a Superpage. Each superpage contains a number of pages (the third level). Data Units and
VisDef can be dropped by a user or be entered from a macro into either pages or superpages. DataUnits dropped at nodes which are not at the lowest level of the trees will be applied recursively down the tree, until they reach the lowermost level.

Each page has a number of children, called Subpages. The subpages will be created by their page parents, as required by the matching rules, and the user does not interact with the subpages directly, but can influence their behaviour by modifying the matching rules. To take a simple example, a GRIB file containing 10 different fields and dropped onto a page which does not contain any data will by default generate 10 subpages.

The leaves of the page hierarchy are the ones which effectively control the visual output, using the concept of a Data Object. A data object is the most atomic type of information handled by PlotMod, consisting usually of a single scalar or vector field or image, or a selected set of observations.

Text and Legend objects are also assigned separately to the Subpage, in order to allow for better control from the user.

Last Update 26 June, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
3.1 Conceptual Design

3.1.6 Matching Rules and Subpage Creation

The subpages are data-dependent, and are created by each page at the lower-most level of the tree, based on the DataUnits dropped onto it (or assigned to it by a macro command plot) and on "matching rules". These rules determine whether two meteorological variables are to be plotted on a single canvas or on different drawing areas. Matching restriction follows some general guidelines which will provide a predictable framework for METVIEW users, and some specific rules for each view.

General Matching Rules

- Rule 1.1 - "All is well that matches well"
  - All the attributes associated to a meteorological variable may be defined a criteria for matching, including: time, level, parameter, variable type and level type. Matching can be enabled or disabled by the user for all supported attributes.
  - By default, METVIEW will assume that if a user has requested to view two different data units on the same window, he has a good reason for it. Therefore, the default action will be to try to find a way to display these different data units together. Only when no sensible course of action is possible, will the user drop be rejected.
- Rule 1.2 - "Allow for non-synoptic hours"
  - A time tolerance will be applied to time matching, to allow for data collected at non-synoptic hours (such as images and observations) to be plotted together.
- Rule 1.3 - "Different Data, Different Colours"
  - For all situations, the default behaviour when plotting two or more data units in the same window will be to use different colours for displaying them, to make comparison easier.
- Rule 1.4 - "Same Parameter, Same Axis"
  - For all axis plotting (such as Vertical Profile, Metgram) there are situations when two different data units are plotted. If the parameter is the same, the axis will be re-scaled to allow for the variation on the two data units.
- Rule 1.5 - "Different Parameters, Separate Axis, Same Scaling"
  - For axis plotting, there are cases when two different parameters are to be plotted together (such as 2m temperature and dew-point temperature). In this case, a second axis will be drawn, which will display the variation of the second parameter. In order to allow for consistency, both axis will have the same scale.

Matching Rules for Applications associated with a MapView

- Rule 2.1 - "One Drop, many fields"
  - When one DataUnit is dropped into a page, each field of a DataUnit is to be drawn on a different subpage.
- Rule 2.2 - "Two or More Drops"
• When two DataUnits are dropped into a page, individual fields from different DataUnits are plotted together if the matching criteria is satisfied.

Matching Rules for Applications associated with an CrossSectionView

• Rule 3.1 - "One Drop, One Time"
  • All data from one drop (contained in the its definition) are sent to the application program for processing. It is the responsibility of the application to verify whether the data is correctly defined for it.
• Rule 3.2 - "One Drop, Different Time-Steps"
  • By analogy with the case of the MapView, as many different subpages will be created as there are different time-steps.
• Rule 3.3 - "Two or More Drops"
  • PlotMod will accept two drops to be plotted together, provided that they satisfy the matching criteria.

Matching Rules for Applications associated with an VerticalProfileView

• Rule 4.1 - "One Drop"
  • If more than one parameter is contained in the drop, PlotMod will create one subpage per parameter.
  • If more than one timestep is contained in the drop, PlotMod will create one subpage per timestep.
  • After this filtering process, DataUnits are sent to the application program for processing. It is the responsibility of the application to verify whether the data is correctly defined for it.
• Rule 4.2 - "Two Drops, same Parameter"
  • By analogy with the MapView, if the parameter in the two DataUnits is the same and other criteria match (e.g., level and time) the Data Unit will be plotted on the same subpage.
• Rule 4.3 - "Two Drops, different Parameter"
  • If two drops with different parameters are dropped in the same window, a second horizontal axis will be created (with a different colour) to display the values of the second parameters. Both axis are to have the same scale. The vertical axis is also re-scaled, as needed.

Matching Rules for Applications associated with TimeView (e.g, Metgram)

• Rule 5.1 - "One Drop"
  • If more than one parameter is contained in the drop, PlotMod will create one subpage per parameter. By definition of metgram, all time steps will be plotted together.
• Rule 5.2 - "Two Drops, Different Time Steps"
  • By analogy with the MapView, if the parameter in the two DataUnits is the same and other criteria match (e.g., level and time) the Data Unit will be plotted on the same subpage.
  • By default, the time axis will be extended to allow for the user to view the two data units together (this is consistent with rule 1.1). For example, a metgram could contain a variable (e.g. 2T) generated by the T-4 forecast and valid for 10 days, plotted together with the same variable generated by the T-1 forecast.
• Rule 5.3 - "Two Drops, Different Parameters"
  • In accordance with Rules 1.1 and 1.5, when two different parameters are to be plotted, a second vertical axis will be created, to display the values of the second parameter.
3.1 Conceptual Design

3.1.7 Layout Definition

The plot layout determines what is seen, not what is plotted. The layout definition consists in specifying the page hierarchy, in the terms of page and super-pages. In the initial version of PlotMod, the specification of the Plot Window will be done by an editor which will support the page definition in the similar way as as the current "SuperPage" command, with the extension that pages can be embedded within other pages, and that a page could have visible subpages, specified in terms of rows and columns.

Therefore, the textual description which is equivalent to Figure 3.1 would be:

```
window_height = 500
window_width = 500

profile_view = view {
    TYPE : P2VIEW,
    TOP_PRESSURE : 1000,
    BOTTOM_PRESSURE : 50,
    POINT : [-90,0])
}

ul_page = page( upper_left, view : profile_view,
    PAGE_X_LENGTH : window_width/2.0,
    PAGE_Y_LENGTH : window_height/2.0)

cross_sect_view = view {
    TYPE : P2VIEW,
    TOP_PRESSURE : 1000,
    BOTTOM_PRESSURE : 50,
    LINE : [-75,0,-75,180]
}

ur_page = page {
    view : cross_sect_view
    PAGE_X_LENGTH : window_width/2.0,
    PAGE_Y_LENGTH : window_height/2.0)

antartica = view {
    TYPE : "XYVIEW",
    MAP_PROJECTION : "Polar Stereographic",
    MAP_HEMISPHERE : "south",
    MAP_VERTICAL_LONGITUDE : 0,
    AREA : [52,45,-43,-130]
}

lower_page = page{
    view : antartica,
    PAGE_X_LENGTH : window_width,
    PAGE_Y_LENGTH : window_height/2.0,
    N_ROWS : 1,
    N_COLS : 3
}
```
plot_window = superpage(
SUPER_PAGE_X_LENGTH : window_width,
SUPER_PAGE_Y_LENGTH : window_height,
plot_start : "top",
pages : [ul_page, ur_page, lower_page] )

The differences between this textual description and the one used currently in the superpage macros are:

- Inclusion of the view directive, which creates new views.
- Association of a view to a page.

This textual description also allow for a backwards compatibility to be attempted between PlotMod and the current macros which use the superpage directive.

In the interactive layout definition, a tree hierarchy will be shown to the user, displaying the current page hierarchy. For each page, the associated data units, views and visual definitions will be presented, allowing the user to interactively modify the contents of the page hierarchy.

Last Update 21 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
3.2 Colour Choice

Introduction

This chapter is still under construction!
3.3 New User Interface

3.3.1 Introduction

The default plot window interface for VisMod is shown below. The comments on specific items follow, with a view of generating a new user interface for use with Plot Mod.
3.3.2 Window Organization

As a whole, it was felt that the current interface does not make an optimal use of the screen for drawing maps. A lot of unused space is taken by the window where the icons are located and by the window which is used to display the cursor location.

A suggestion which has been made is to reorganize this window in such a way that:

- The icons appear on a window below the menu window, organized horizontally.
- The window which is used to display the cursor position is no longer used.
- The message area is made smaller, and a sliding bar is made available to loop through messages.

3.3.3 Icons

The issues most mentioned by the users regarding this interface were, as regards the icons, had to do with the fact that the icons do not correspond exactly to the actions performed by VisMod.

- The icons portrayed as lenses in reality correspond to two very different things: The magnifier icon and the zoom icon should be made different, since the second one effectively corresponds to a new area definition.
- It was suggested that the new "zoom area" icon would be similar to , which would be put alongside the current icon , which corresponds to the magnification.
- There seemed to be an overlap between the icon used for showing grib values ( ) and the one for selecting point locations ( ). In practice, the two icons have to be combined. Since the default mode for the cursor (as selected by the icon) is a "point location" mode, the point icon was felt to be redundant. From that, it follows that the grib value output facility could be reduced to a single icon. Alternatively, the grib value would be shown in the top level window, alongside with the (lat, long) coordinates, as the default behaviour (which could be turned off).
- The majority of users had never used the "area select" facility (indicated by icon ). This perhaps could be explained by the limited number of applications which make use of this facility.
- Due to the fact that having a screen preview is part of the normal working routine (partly because the current VisMod is not fully WYSIWYG, partly because the are inherent differences between screen and paper), it was requested that the Postscript preview icon is moved to the main Plot Window interface.

As a more general issue, there should be a menu equivalent to every icon on the PlotMod interface (the same comment applies to the GenApp interface), as prescribed by the "OSF/Motif Style Guide". Additionally, each icon could be associated to a text "bubble" which briefly describes its content (similar to the Microsoft Word interface).

Last Update 23 March, 1997 by Gilberto Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts/ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
3.3 New User Interface

3.3.1 Introduction

The default plot window interface for VisMod is shown below. The comments on specific items follow, with a view of generating a new user interface for use with Plot Mod.
3.3.2 Window Organization

As a whole, it was felt that the current interface does not make an optimal use of the screen for drawing maps. A lot of unused space is taken by the window where the icons are located and by the window which is used to display the cursor location.

A suggestion which has been made is to reorganize this window in such a way that:

- The icons appear on a window below the menu window, organized horizontally.
- The window which is used to display the cursor position is no longer used.
- The message area is made smaller, and a sliding bar is made available to loop through messages.

3.3.3 Icons

The issues most mentioned by the users regarding this interface were, as regards the icons, had to do with the fact that the icons do not correspond exactly to the actions performed by VisMod.

- The icons portrayed as lenses in reality correspond to two very different things: The magnifier icon $\mathcal{M}$ and the zoom icon $\mathcal{Z}$ should be made different, since the second one effectively corresponds to a new area definition.
- It was suggested that the new "zoom area" icon would be similar to $\mathcal{E}$, which would be put alongside the current icon $\mathcal{E}$, which corresponds to the magnification.
- There seemed to be an overlap between the icon used for showing grib values ($\mathcal{P}$) and the one for selecting point locations ($\mathcal{N}$). In practice, the two icons have to be combined. Since the default mode of the cursor (selected by the $\mathcal{R}$ icon) is a "point location" mode, the point icon was felt to be redundant. In that, it follows that the grib value output facility could be reduced to a single icon. Alternatively, the grib value would be shown in the top level window alongside with the (lat, long) coordinates, as the default behaviour (which could be turned off).
- The majority of users had requested the "area select" facility (indicated by icon $\mathcal{A}$). This perhaps could be explained by the limited number of applications which make use of this facility.
- Due to the fact that having a screen preview is part of the normal working routine (partly because the current VisMod is fully WYSIWYG, partly because the inherent differences between screen and paper), it was requested that the Postscript preview icon is moved to the main Plot Window interface.

As a more general issue, there should be a menu equivalent to every icon on the PlotMod interface (the same command applies to the same interface), as prescribed by the "OSF/Motif Style Guide". Additionally, each icon could be associated to a text "bubble" which briefly describes its content (similar to the Microsoft Word interface).

Last Update 23 March, 1997 by Claudio Cámara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts ECMWF, Centro de Previsão de Tempo e Estudos Climáticos (CPTEC/INPE).
METVIEW - Visualisation and Plot Module

4. System Interfaces

Introduction

One important concern on the development of PlotMod is the need to preserve, to the maximum extent possible, compatibility with the exact behaviour of VisMod, relative to the other METVIEW modules. In other words, the impact of the substitution of VisMod by PlotMod for the other modules should be as small as possible. With that aim in perspective, the next three sections examine the interaction between different VisMod and the existing METVIEW modules.

4.1 METVIEW Requests Processed by PlotMod

Section 4.1 provides a general description of the requests received by PlotMod and indicates the actions expected in each situation.

4.2 Detailed Examples of METVIEW Requests

Section 4.2 provides a step-by-step presentation of some typical requests processed by PlotMod.

4.3 Relation to PlotMod and Magics

Section 4.3 provides an analysis of the relationship between PlotMod and Magics.

Last Update 1997 by Giuseppe Scarpetta. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts (ECMWF).
ETVIEW - Visualisation and Plot

Module

4.1 METVIEW Requests Processed by PlotMod

Introduction

The primary means of communication between PlotMod and the other METVIEW modules is by means of request-and-reply mechanisms. The METVIEW modules which communicate with PlotMod are METviewUI (the main interface), macro (the macro player) and MagProc (the interface to Magics). The user's data processing actions by the user will also cause requests to be generated, as explained below.

There are six different situations which PlotMod processes and generates requests:

- Initialization of a new plot window by the user.
- Dropping of a file onto a plot window by the user.
- Response to request actions generated by MetviewUI.
- Interactive action or command generated by Macro.
- Editing of plot window coordinates for METVIEW applications.
- Messages reported by MagProc.

Please note that in this discussion the term "plot window" refers to a generic plotting device which may be a separate window on a paper plot. Also, it may correspond to a normal plot window or to the notion of superpage as defined by the SUPERPAGE command.

Initialization of a new plot window

Since plot windows are independent of each other, we shall consider the requests that generate a new plot window separately. These requests are produced in the following situations:

- The user requests to open a new plot window either by double-clicking a plot window icon, or by requesting a new plot window from the icon menu (PLOTWINDOW).
- The user requests to view a new plot window of a data unit (GRIB and BUFR).
- MetviewUI requests to open a new plot window from a request to view the result produced by an application (CLEAN).
- Macro initiates the processing of a new plot window using TWINDOW and SUPERPAGE.

In summary, the requests which initiate a new plot window include the PLOTWINDOW and SUPERPAGE directives and the CFILE, BUFR and CLEAN (when not preceded by DROP).

The reaction of PlotMod to these requests will be:

- A new plot window is generated with the appropriate background.
- If there are any data units associated with the request, the matching rules will be applied to create the necessary data for display into.
- The data is fetched using module (MetviewUI or macro) giving the identifier associated with the parameter passed by the user, called DROP_ID, is used for all further requests which deal with this window. This holds both for interactive and batch use.

User Drag action

When a user drops a file onto the window, MetviewUI responds to a drop by generating a request about the identity of the window which has received the drop (this is parameter ID, along with the identifier for the window. PlotMod is informed of the drag-and-drop operation and matches the icon's file extension.

MetviewUI responds to a drop

After receiving the request, MetviewUI decodes the window's identification, MetviewUI will send a DROP request back to PlotMod. The request contains the ID corresponding to the desired plot window, along with specific data about the dropped file. If the request is generated by an application, a CLEAN request is issued which provides information about the application.

PlotMod's response consists of the following steps:

- decode the dropped file
- use the settings in the file
- plot the data, and
- reply to user that the request has been completed (or that an error has occurred)

Macro Batch Interface commands

The communication between PlotMod and Macro follow the same pattern as the above-mentioned response to a drop action. It is understandable, since a macro is expected to be able to reproduce all actions performed directly by the user. Therefore, the macro will generate a DROP request (DROP REQUEST) and/or VISDEFs (e.g.: GRIB, VECTOR FIELD or MATRIX) and/or data units (e.g.: GRIB, VECTOR FIELD or MATRIX) and/or VISDEFs (e.g.: GRIB, VECTOR FIELD or MATRIX).

Plot Mod's response is the same: decode the data, match it against the current setting, plot the data, and inform the user of the results.

Providing Mandatory Interactivity

Many applications (VIEW, CROSSSECTION, and VIEWMODEL) require a geographical location (area, point, etc.) as input. In the editing window associated with the application, the user may ask for a location which makes him with a geographical map (in cylindrical project) where he would like to place the location eventually.

This situation leads to an ADD X request, which is sent by MetviewUI to PlotMod. This request initiates a Map Editor Interface to the user, who makes his choice. PlotMod then sends a MESSAGE-INPUT request which informs the desired
<table>
<thead>
<tr>
<th>Situation</th>
<th>Input Request</th>
<th>Action</th>
<th>Reply Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Plot Window - click plot window</td>
<td>PLOTWINDOW, GRIB, SUPERPAGE, CLEAN</td>
<td>Match Data, generate a new Plot Window and plot the result</td>
<td>REPLY-DROP (with DROP_ID)</td>
</tr>
<tr>
<td>New Data or VisDef - sent from event</td>
<td>DROP-GRIB, DROP-CLEAN, DROP-PCONT (and similar ones)</td>
<td>Match data, change view (if necessary) and plot the result</td>
<td>REPLY-DROP</td>
</tr>
<tr>
<td>Map Coordiater - user tries to increase a map</td>
<td>INPUTWINDOW</td>
<td>Provide a map interface and obtain area, line or point coordinate</td>
<td>MESSAGE - INPUT</td>
</tr>
</tbody>
</table>

Last Update 2/7 by Camara and Fernando. Copyright 1997(c) European Centre for Medium-Range Weather Forecasting, ECMWF.
4.2 Resolve to use a PlotMod and GenApp
4.3 Relations between PlotMod and MagProc

Introduction

The user is still under construction!

Last Updated 1997 by Câmara. Copyright 1997(c) European Centre for Medium-Range Weather Forecasts (ECMWF).
5. Software Implementation

5.1 Implementation Techniques

5.2 Representing the Hierarchy

5.3 Allocation of Organization for DataUnits and VisDef

5.4 Processing Requirements

5.5 Supporting Different Graphics Engines and Devices

5.6 General Structure of PlotMod Classes
5.1 Software Implementation Guidelines

Introduction

Metview development is expected to continue over many years, and more features and functionality are being implemented. It is important to keep the system in mind, it is very important to use software implementation techniques which make maintenance and additions to PlotMod. Two principles are paramount: using a good documentation and developing flexible modules.

Documentation

There are two sources of information on PlotMod: this document and the code itself. Instead of providing a separate document describing the code, it was considered that the code description should be made part of the document as it should evolve together. For that purpose, all the class headers are included in the code, with the corresponding description of the class as a whole and of each individual function. In this way, each class is described in terms of the CRC (Class-Responsibility-Collaborators) model.

The CRC for each class is given in terms of responsibilities, with each view of a class corresponding to the one being described as referred to as the class-oriented software development as the activity of design and implementation consistent with the components, which can be combined to form a complex system.

Implementation

In the development of PlotMod, a specific coding style has been pursued, in order to lighten the load of maintenance of the code. Some guidelines have been applied to the maximum extent possible:

- All class names are in uppercase.
- The C++ naming conventions have been used whenever applicable.
- The documentation of the source is provided in a readable format, by means of external definition tables.
- The documentation has been applied to all PlotMod classes.
- The C++ code is shown dynamically, by means of external definition tables.
- The C++ code is shown dynamically, by means of external definition tables.
- The C++ code is shown dynamically, by means of external definition tables.

Coding

The coding style is as follows:

- All methods are defined in uppercase, without underscores, e.g.
Use of Template Library

The STL is one of the most important recent additions to the tools available to C++ programmers. This is for good reason. For example, the class template can be used for manipulation of user-defined arrays.

Furthermore, this has been accepted as part of the new C++ standard.

External behaviour

One of the characteristics of a large system in general is the sheer diversity of requests, which are not only handled in significant areas (such as PLOTPAGE, SUPERPAGE), data (including IOFLAGS), or graphical representation (such as POOLPOINTS, visual definitions (PCONT, PWIND), to name a few.

This diverse collection of requests which a significant part of the code is spent inquiring about, and then performing an appropriate action. The code would look like this:

```c
if (strcmp(command, "PLOTPAGE") == 0) {
    // PLOTPAGE logic
}
else if (strcmp(command, "SUPERPAGE") == 0) {
    // SUPERPAGE logic
}
... // other cases...
```

In order to reduce code repetition, we have attempted to concentrate most of the information related to frequently occurring requests into PlotMod on an external file.

The cases are grouped into associated by PlotMod and the actions associated to each request are contained in external file, `PlotModTable`, which is easily understandable and which allows easy modification of the program behaviour.

For example, in the `Create` command, the SUPERPAGE command, indicates:

```c
class CREATE, ...
BUILD: SuperPageBuilder,
... // code...
```

The above code indicates that the `SUPERPAGE` command is associated to the "Create" action, which is what we would wish. Also, that it uses the "SuperPageBuilder" to build the tree hierarchy, which is meaningful (since a superpage contains no data). In combination with this, there is no code (as defined below), a significant economy on both code size and complexity.
Programming with Prototypes

The base class is classed as a "prototype" of the class. One of the requests are coded using a C++ idiom called "prototyping" [C++92]. This idiom allows dynamic reconfiguration of the derived class usages in a similar way in the MetviewUI (GenApp).

One of the most important benefits is how to hide the dependency between the base class and the client. Suppose that PlotMod has a base class Builder which takes an abstract window and a command which has to deal with various types of commands.

A "command" could be

```cpp
BB P
uperPa = new (tree);
command = WINDOW;
BB P
omer (tree);
BB P
er (tree);
BB P
o on;
```

The "ables" places:

```cpp
-- Builder take (command);
(command);
```

The "able" derived which implements the specific builder for each type of command

```
Builder . first ...
main program (main.cc) is performed. The program then
process ...
ast abstract type of builder (which is performed by the Builder:Make
methods
en, there are two important benefits:

- can be changed with subclasses (such as Builder), and all the subclass information
- can be changed dynamically, without the need to recompile the base class or
```
5.2 Page Hierarchy

Given the two-dimensional nature of PlotMod, it is most important to devise an adequate representation of the page hierarchy described in section 3.1.5. A page hierarchy, which consists of different types. From a software design point of view, it is very useful to group objects (pages, subpages, views and data units) uniformly. To achieve greater code neatness and code cleanliness, PlotMod relies on a general abstraction, which we will call "Pageable." This class represents both the bottom-level structures of the page hierarchy and their containers (pages and subpages).

The hierarchy is encapsulated by methods for inserting, removing and drawing objects on the page, and for obtaining parents and children on the page hierarchy tree. Each of these methods is defined as an abstract function (in C++ terms, a virtual method), which will be implemented in the concrete function of its subclasses.

In terms of "Design Patterns," the Pageable class is an example of the Composite pattern. A typical use of this pattern is for representing objects into tree structures, such as a UNIX file system. A UNIX file may contain other files or other directories.

A brief description of each function of the representable class follows:

- The insertion operation will determine the object and its children. For a subpage, this operation might require traversing the hierarchy and call a Draw operation for all its children. For a view, it will require the graphics engine to perform the required graphics.

- The removal operation will remove new branches and leaves it the tree. Note that only the representation of the page can remove their children. This operation is defined at a higher representation level in order to ensure greater transparency on the resulting output. See the "Design Patterns" book (page 107) for a specific discussion on the
This pattern for representing the PlotMod page hierarchy:

- structures have
- objects are
- each class is
- the description
- a special case of

Each hierarchy is

This is used at an

involved

Two areas are of

example

views into

To the entire plot window. This will amount to asking
to most le
be a ship.
mime -obj e
ve (or exam
net to the gra
DataUnit (R)
DataUnit
salient to de
cracy insert
im; Fields v
There is a general term
until how it
monot idea

Last 1
for M

ara and Fernando Li. Copyright 1997(c) European Centre
Wf.
5.3 Data Base: Organizing Data Units and Units

Data Defs

In order to code the met data, it is essential to identify the basic data def. As a general rule, in each data unit, there is a primary data def, the def that represents the whole data unit and from which the various data def in a data unit are derived. Data def in a data unit are often related at a declarative level through a hierarchy that is the same as the def hierarchy in METVIEW. The def hierarchy is a means of organizing data def at a semantic level. Data def in a data unit are related at a procedural level through the use of the visdef directive. The visdef directive is a means of organizing data def at a procedural level. Thevisdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level.

Data def hierarchy

In order to code the met data, it is essential to identify the basic data def. As a general rule, in each data unit, there is a primary data def, the def that represents the whole data unit and from which the various data def in a data unit are derived. Data def in a data unit are often related at a declarative level through a hierarchy that is the same as the def hierarchy in METVIEW. The def hierarchy is a means of organizing data def at a semantic level. Data def in a data unit are related at a procedural level through the use of the visdef directive. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level. The visdef directive is a means of organizing data def at a procedural level.
The current object is considered a "data unit" (an internal METVIEW interface icon, which 
takes an "visdef" value. Such is the approach 
when both entities. Each icon contains a copy of the 
its origin as a PCNT request, so that all relevant information can 
all relevant information can be used to simplify operations, the MvIcon class is reference 
counted to the same memory positions. The deletion of an 
object which is implemented as a PCON request. In PlotMod, the reference counting is 
deleted. In PlotMod, the reference counting is 
implemented using a 
implementation described in Coplien [1992].

Each base is a

---

Entity linking

Given an entity-relation data unit and visdef relationship in entity-relationship terms, data units and visdefs are 
are illustrated by the Figure 5.3.1.

There are various 
alterations can be used in different circumstances. In this approach, for 
each alteration about the data unit, visdefs in a data base, and to provide various 
alterations can be used in different circumstances. In this approach, for

This is a list of all entity units which are used to display many data units at one. This would 
not necessarily be a single, centralized place where the information

There is a list 
that stores:

- a list of all visdefs (implemented as STL list). The main
- a list of all visdefs (implemented as an STL multimap).
- a list of all visdefs (implemented as an STL multimap).

In relation 
which are used on all levels of that superpage.
Figures 24 25: Misdels and presentables.

Last update: August 30, 1998

Copyright 1997(c) European Centre for Malaria and Filarial Lymphatic.

VF.
7 - Visualisation and Plot

5.4 Visualisation Tests

Overview

The application programming interface (API) associated with PlotMod, described in Section 4.1, has led to the following actions which are performed:

- Creation of data units and superpages.
- Decompression of data units and visdef into plot windows.
- Insertion of plot windows into pages.
- Insertion of pages.

Each of these actions is handled by the PlotMod class (CreateAction, DropAction, VisDefAction), which is called by the main METVIEW module depending on the current context.

Drawing

Out of the various API functions, those related to drawing are the most important, as they lead to the generation of graphical output and visualisation options, and will be examined here in more detail.

In the context of the METVIEW programming environment, drawing consists of four main phases:

- The window's page tree hierarchy.
- The device driver.
- The graphics library.
- The data unit.

As an example, consider a data unit containing a data unit is requested to be "visualised" by the user. This could be a GRIB request. PlotMod will open a new superpage, then decode and render the data unit and subpages are required to store it. After that, the graphics library is used for the actual drawing.

Taking advantage of the fact that drawing classes are part of PlotMod, implemented as abstract base classes such as PlotWindowBuilder, SuperPageBuilder and

- MapViewMatcher and VertProfViewMatcher.
- Vis5D and OpenGL.

as well as
The interaction for the diagram is depicted on Figure 5.4.1, which shows the interaction diagram for constant cases. In essence, the flow of actions shown by the diagram is as follows:

1. The request and calls a Builder, to build the new action object.
2. Another object matches the data units and visualises the data in the plot window.
3. A drawing message is sent from the PlotModAction to the Plot hierarchy. In turn, this branch (a page or another hierarchy) will contain functions for visualisation of the section and will call graphical primitives from the graphical drawing of XWindow, OpenGL, and Postscript.
Visualisation and Plot

5.5 Plotting

Displaying Different Graphics Drivers

The current choice of graphics engine, and MAGICS will continue to be the main plotting engine. Different graphics engines, including the Vis5D Visualizing software, the GraphicsEngine class encapsulates the general behaviour of the graphics engine and therefore, the other PlotMod class deal with this abstract concept that the graphics engine is MAGICS, Vis5D or

---

Copyright 1997(c) European Centre for...
6. Conclusions and Acknowledgments

Conclusions

Acknowledgments

The authors would like to acknowledge the important contributions to this conceptual design:

- the overall hierarchical design (item 3.1.5) is by Baudoin Raoult, who has given invaluable support.
- the design of the Views and Data model (item 3.1.4) and the matching rules (item 3.1.6) is by Vesa Karhila and Jens Daabeck.

Under construction!