A Java-Based Meteorological Workstation

Hans-Joachim Koppert – Deutscher Wetterdienst, Kaiserleistr. 42, 63067 Offenbach, Germany
Hans-Joachim.Koppert@dwd.de

1. Introduction

The architecture of the meteorological workstation systems in operational use at DWD dates back to the early nineties. These systems were built on C, X/Motif and OpenGL. Their basic functionalities are pretty similar, but have to be maintained separately. In order to unify all the functionalities in one single system, to implement new features, to gain flexibility, and to optimise maintenance we have decided to replace the old systems with a new architecture based on the Java Programming Environment.

2. The project

The system is now called NinJo. NinJo is at the time of writing a joined effort with MétéoSuisse, DMI (Danish Meteorological Institute), GMGO (German Military Geophysical Office), and DWD. The project started in December 1999. The requirement specification was available in summer 2000. Work on the overall design and the evaluation of the technologies we wanted to utilise was completed late spring 2001. During this stage of the project, certain disposable prototypes were used to highlight several architectural issues. Since then we are in the process of refining the use cases, refining the design and implementing the basic framework. Work on first modules implementing the desired operational functionalities building upon the final design has started in autumn 2001.

3. The goals of the NinJo project

Since NinJo is a joined project, it is very important to have clear, open, and expandable software architecture. NinJo has to be independent from hardware and operating systems, because the organisations involved rely on different IT-infrastructures. Although there are basic data sets with their respective use cases like Synop, Ship, Temp, NWV-data, and so on that are used by all the organisations involved, there are numerous data sets and applications that are specific to the partners. Therefore special attention is paid to generic frameworks that allow to independently implement applications. This includes the integration of new layers into the client's framework as well as the access to different data supply and backend systems, data communication and middleware infrastructures.

4. Software Architecture

4.1 Overview

Java was chosen as the computing platform, because it was the only environment that was able to fulfil our requirements, especially the independence from the operating system. Furthermore it offers a rich set of APIs like Java2D, Java3D, JAXP, and Java Advanced Imaging, only to name a few. The APIs offered by the available Java Development Kits cover most of the necessary functionalities for building a complete Meteorological Workstation System. Over the years Java has become a mature platform that constantly enhances its functionalities and performance.

Figure 1 shows the three tiers of the NinJo architecture. The standard client at the weather forecasting offices will be a Fat Client. The servers will offer data and computing services not only to Fat Clients but also to Thin Clients and batch applications. This concept allows unification of most of the meteorological post-processing within the NinJo system. The data tier is open and flexible and will offer a high performance flat file system as well as access to RDBMS.

4.2 The NinJo Client

The design of the Client framework uses a design pattern known as PAC (Presentation Abstraction Control). The PAC design is actually a hierarchy of MVC (Model View Controller) triplets. This PAC-Framework allows managing layers, processes events and user interactions. It also guaranties that all components are decoupled and therefore plugable. The PAC-Framework is configured through XML. This configuration allows building of very simple clients with only a radar display or really complex applications with 3D and interactive map production.

The NinJo client has one main window for the display of data in a geographical context. This main window has one main scene and up to 3 secondary scenes and several secondary windows for the display of meteograms, soundings and so on. The client will allow seamless integration of 2D and 3D-visualisation. Switching between

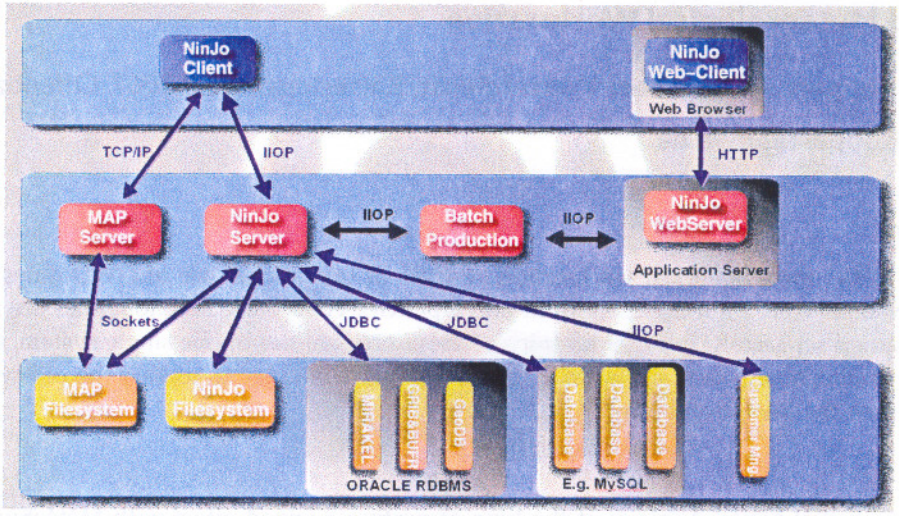

Fig 1: The architecture of NinJo

these two display modes will be made easy by preconfiguring all the necessary settings of 3D-views and attributes. Figure 2 shows one possible client configuration with a main scene depicting surface observation on a high-resolution geographical background, a secondary scene showing temperatures over Europe and another secondary scene showing a 3D isosurface of the liquid water content.

The visualisation in NinJo is based on an API called GOF (Graphics Object Factory), a scenegraph API that abstracts from the actual graphics APIs like Java2D, Java3D, and OpenGL. The GOF handles the geometry and the appearance of graphical objects. Since NinJo will also replace our batch production of meteorological charts and images, we will not just focus on the display of data on the screen, we will also implement off screen rendering for vector based metafiles like SVG, Flash, and Postscript.

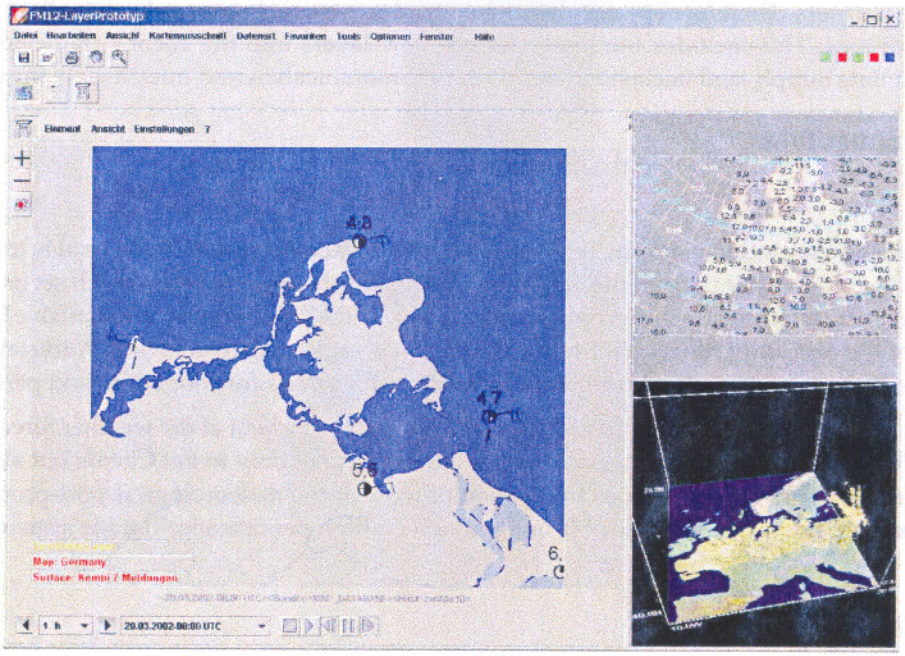

Fig 2: The NinJo Client

4.3 *The NinJo Server*

The server tier offers several ways of accessing and storing data. Due to the system's open architecture the project partners can adopt the system to their needs, e.g. distributing services and data storage according to their local hard- and software infrastructure. At DWD we will store and process data that is used heavily in operational forecasting operations at regional offices in flat file system. Archived data and data that is not used daily, like high-resolution GIS data, will be made available in a RDBMS at our central office.

Data decoding, which is currently - in a parallel project - also ported to Java, can be performed centrally or locally according to the requirements of the respective project partner.

In order to make NinJo as scalable and fault tolerant as possible and to have the additional benefit of interfacing with "non-Java-software", DWD has chosen CORBA over RMI as middleware. The middleware is accessed through an abstraction layer allowing use of different CORBA implementations or even of RMI.

The servers themselves consist of several services for different data types like point data, gridded data, configuration data, revised forecasts, or image data. The servers also host print services, computing services, or decoding services. All data servers follow one generic design to facilitate the implementation of new data types and services

5. The Status of the Project

With the NinJo project we were able to join resources from several projects and organisations. The bundling of forces together with support from IT-consultants ensures that we will be able to deliver the system in time. It is planned to have rollout in mid/late 2004.

Since it is an international project with development sites in Offenbach, Zürich, Copenhagen, Potsdam and Traben-Trarbach, we had to set up an infrastructure for collaboration with a common code and document repository. Besides basic training courses on Java and CORBA, the project team itself carries out training courses on the basic NinJo frameworks (client layer framework, Graphics API...). These courses enable team members to implement the requested meteorological applications. We have decided to implement the system in half-yearly design-implementation cycles. In this phase we are refining the use cases, building a detailed design, implementing the requested design utilising the basic NinJo-frameworks, and performing quality assurance. The resulting versions are reviewed by an evaluation group of experienced meteorologists and forecasters.

The functionality of the versions is as follows:

Version	Release Date	Feature List
0.1	Oct. 2001	Graphic-API 3D-visualisation of isosurfaces 3D-textured DEM-data
0.2	Feb. 2002	Client framework Surface observation layer (FM12 only) with limited functionality Geovector layer based on VMAP0 and German high res data Access layer
0.3	July 2002	Gridded data server and client layer Configuration framework based on XML Logging and error-handling API
0.4	Winter 2002/2003	Raster image framework Satellite data Georaster data Framework extension for secondary windows Additional data services

With release 0.4 we will be able to process and visualise all the basic meteorological data sets. 2003 will see the design and implementation of graphical editing, tools for analysis, nowcasting and warning operations and the completion of the Java-based decoding software.