11.1 HOW TO MAKE AN INTERNATIONAL METEOROLOGICAL WORKSTATION PROJECT SUCCESSFUL

Hans-Joachim Koppert [1]
Deutscher Wetterdienst, Offenbach, Germany
Torben Strunge Pedersen
Danish Meteorological Institute, Copenhagen, Denmark
Bruno Zürcher
MeteoSwiss, Zürich, Switzerland
Paul Joe
Meteorological Service of Canada, Toronto, Canada

1. INTRODUCTION

The Deutsche Wetterdienst (DWD) together with the German Military Geophysical Service (GMGO) started a new Meteorological Workstation project (NinJo) in 2000. Soon thereafter Meteo-Swiss and the Danish Meteorological Institute (DMI) joined the project. Just recently, the Meteorological Service of Canada (MSC) decided to become a member of the NinJo consortium.

The aim of the project is to replace aging workstation systems (Kusch 1994, Koppert 1997) and to provide an unified environment to support forecasting and warning operations. Besides the development of new, or the integration of recently developed applications, it is necessary to provide most of the capabilities of the existing systems. Forecasters are conservative and often would like to stay with their old tools. So we are facing an enormous pressure to implement a system that does (nearly) everything better.

Since NinJo is a multi-national project with 5 partners, diverse hardware and software infra-structures, distributed development sites, and local meteorological needs, a strong requirement is to built the software with a sound software architecture that could easily be adapted to the needs of the partners. But it's not only the software architecture that assures the success of an international workstation project, it is also the management of the project. The management has to make sure that all requirements are incorporated, that resources are effectively used and the communication among the sub-projects is functioning effectively.

2. THE NINJO WORKSTATION

User requirements were collected and structured at the very beginning of the project. These high level requirement and specifications are the basis of the agreement between the NinJo partners. In the course of the project, they are refined when the respective sub-project work-packages are initiated.

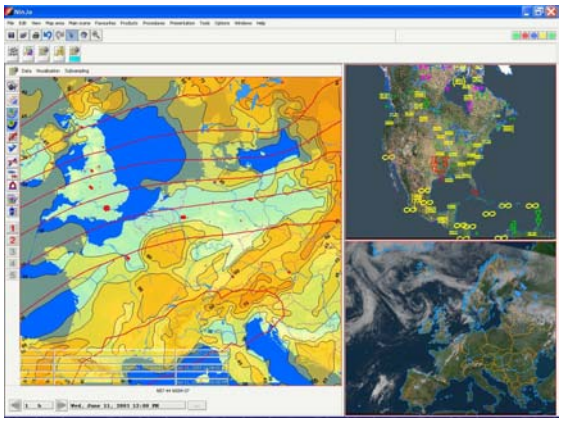

Figure 1: A NinJo window with 3 scenes: gridded data (left), surface observations (top right), satellite image (bottom right)

The following list shows NinJo's main features :

- Integrated 2D and 3D visualization of point data including: surface observations, soundings, lightning data; gridded data; satellite and radar imagery; geographical data; and bulletins
- Batch chart production
- Data decoding and management
- Graphical editor
- Data modification - both point and gridded data
- Monitoring and alerting for observations and NWP data

[1] *Corresponding author address*: Hans-Joachim Koppert, Deutscher Wetterdienst, Kaiserleistr. 42, 63067 Offenbach, Germany; email: Hans-Joachim.Koppert@dwd.de

- Support for multiple windows with data layering, multiple screens, and animation

Expected areas of operation: Guidance, warning/consulting, research, climatology, reports, commercial and private customers

3. RATIONALE FOR WORKING TOGETHER

The NinJo partners have a very long history of developing meteorological applications. Based on this history, there are different rationales for working together on a large and complex project like NinJo.

At DWD and GMGO, several Meteorological Workstation Systems were developed from the early nineties. These Systems became difficult to maintain and upgrade. Furthermore, they lack required features such as GIS support, advanced analysis functionality, data modification modules, seamless integration with 3D visualization and interactive chart modification into one single application. In 1999, a strategic decision was made to replace these aging systems and to move away from dedicated hard and software environments. In order to broaden the project's pool of expertise and to strengthen the resources, DWD and GMGO happily welcomed DMI, MeteoSwiss and MSC as new partners

The DMI started its first meteorological workstation project in 1990. Based on a cost / performance evaluation and the money available DMI selected SUN Sparc workstations and started to develop applications. With very limited man power, an approach was chosen where one application was developed for each data type facilitating fast operational deployment to forecasters who at the time had no such tools at all. During the 90's, DMI moved from Sparc to Intel based Solaris OS workstations and then started to move towards Java. After a decade with workstations, DMI saw a large part of the development resources going into maintenance, an uncertain OS and hardware future, a requirement for data integration in the applications.

With the resources available, the options were to port existing applications (most still in Motif) to Linux or acquire a "new" system that could provide the data integration (e.g. Horace). Developing a new DMI system from scratch was out of the question. With the option of participating in NinJo, it was easy to reach the conclusion for the DMI to make a final Solaris Intel software release, join NinJo and be part of a real meteorological workstation project.

The meteorological software currently used at MeteoSwiss consists of many, rather small and isolated applications which were increasingly difficult to maintain. In the past decade, there have been efforts to develop an in-house integrated solution, but there were too few resources available to accomplish this task in a reasonable time. Embedding commercially available systems into the existing hardware and software environment would likewise be difficult to achieve. However, the participation in the NinJo project provided MeteoSwiss with a long needed, integrated meteorological workstation.

Similarly, the MSC has tried to build/buy/develop an integrated workstation for many years to replace aging, inadequate systems while including new science and technology. It is a big investment, a high risk venture, a difficult project to manage and very difficult to clearly state the requirements and specifications A new driving impetus for the MSC was the restructuring to fewer offices which assumed that the new and maturing science and computer technology - models, algorithms, visualization, software, processing, telecommunications, data storage – could be leveraged (GEM refer; Scribe ref; Joe et al, 2002; FPA).

The advent of affordable high power graphical workstations in the late eighties led to the development of many independent but very effective forecast applications. The code was difficult to maintain or upgrade because they were designed for specific applications and situations and lacked a software or data architecture or an overall concept that would enable it to evolve. Even newly developed radar software, such as CARDS (Lapczak et al, 2001; Joe et al, 2002), soon hit data access roadblocks. So, a well designed software architecture and data infrastructure is needed.

Regardless of which system was the starting point for development, it would require substantial modification and development to meet the MSC vision. So, the driving , requirement was to develop from a solid IT and data foundation. Unique in MSC's IT assessment of software was the NinJo system - it was being designed based on sound software principles -. functional components were isolated from each other and there was an encompassing architecture

4. IT SUCCESS FACTORS

The philosophy of NinJo is to provide a building kit in order to assemble tailored applications from NinJo components. A flexible programming model is needed to allow NinJo partners to develop their own application without hampering each other. With this concept, it is possible that each partner can easily adapt NinJo to their own needs.

Before we discuss the key design principles of portability, client layer patterns, generic frameworks, and easily manageable data sources, we have to explain our NinJo concept of a generic Meteorological Workstation. This concept is one of the keys to success!

4.1 A Generic Meteorological Workstation

A generic Meteorological Workstation should be able to run easily at any meteorological institute

of the world. It doesn't assume or prescribe any hardware or data infrastructure. It relies solely on standards for input and output of data and/or products. WMO codes like Synop, Temp, Metar, BUFR or GRIB (to name a few) should be able to be ingested as well as any vector or image format. It is not possible to interface with all back-end systems immediately, but, the architecture of the workstation should be capable of implementing the connection with reasonable effort.

A multi-national Meteorological Workstation can be successful only if the technical pre-requisites of a generic workstation are addressed early in it's development.

4.2 Portability

The NinJo partners run their current systems on a variety of hardware with different operating systems. When NinJo becomes operational, we expect at least 3 operating systems to be used: Linux, Solaris, and Windows. In the very beginning of the project Java was chosen as the computing platform. It promised to be independent from the underlying hardware and from the operating system. Until now we haven't encountered any serious problems with Java's portability. One of the problems that one might face is the availability of certain APIs. As an example on the Mac OS, there is no Java3D nor Java Advanced Imaging (JAI), although the latter could be used without native acceleration.

The main concern with large Java systems was performance. The evaluation of mission critical components of the Java platform was done in detail early in the project (Koppert, 1999). We found that performance is not an issue, even numerical computations can be done in Java. The decision to go with Java meant that we couldn't reuse anything, everything had to be rewritten from scratch including peripheral modules like decoding.

4.3 Building Client Applications

Every NinJo partner wants to be able to build their own versions of NinJo, ranging from simple satellite viewers to full blown workstations.

We call a frame, that displays data, a scene. A scene can display data in several stacked layers. Layers are visualization applications of point data, gridded data, imagery, GIS-data and so on. The configuration of these layers can be done during run time or can be loaded from XML-configuration files. It's obvious that the configuration framework is another key component for success. Everything that is "configurable" can be made persistent. This framework makes the implementation of "favorites" and the "undo" functionality possible.

The NinJo project had to provide a framework that allows the construction of a NinJo client. The design pattern used to fulfil these requirements is called PAC (Presentation Abstraction Control). PAC is a hierarchy of co-operating agents following the Model View Controller (MVC) design pattern. This agent hierarchy is configurable through the above mentioned configuration framework. There is a strict decoupling of the agents, allowing the layers to be independently developed. Each layer manages it's own data. The view of the layer provides GUI-components to be integrated by a top-level agent, the "master of the scene". This top level agent (called the Layer Container) is also responsible for global event handling like the changing of the geographic or the time context.

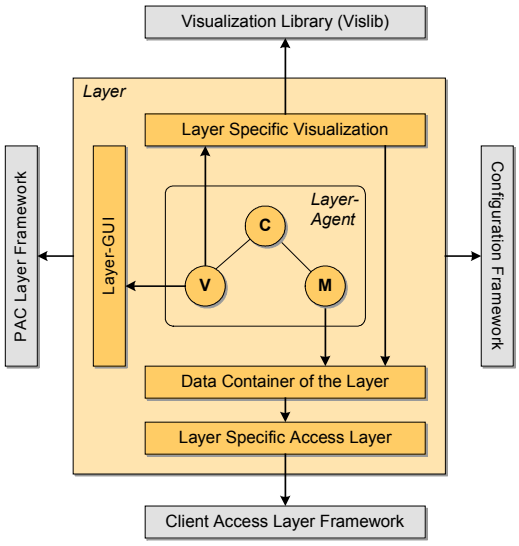

Figure 2: The client architecture with PAC Framework, MVC driven layer agents, configuration framework and client access layer

With this approach, it is feasable for every NinJo partner to develop their own layer independently, if the existing layers are insufficient.

In an international project, it is also necessary to consider internationalization. Java offers an internationalization API that allows NinJo applications to be localized quickly.

4.4 Managing Data Sources

The client access layer (CAL) is the only entry point for a NinJo client to access any meteorological data. This framework is designed in a generic way, such that it handles arbitrary types of data (like gridded data, observational data etc.). It also manages various kinds of data sources, like data servers or local file systems.

All data sources have to provide information about their current contents in a well defined way. This is done in a meta data tree which is similar to a directory tree where a leaf in this tree can be identified with a particular data set from a particular data source

Data sources are required to implement the same service interface for similar data types. It defines all “getter” methods to access data of this type.

If a NinJo client wants to work with a specific data type, the client has to query the CAL for the contents of the corresponding meta data tree. By this means, the client is able to set up a request (i.e. to define the meta data) for a data set and submit it together with additional access parameters to the CAL. All data sources matching the request, are determined and the load-balancing and fail-over strategies decide which of the data sources will be queried for the data.

Since the data source implementation class is configurable, it is possible to write adapter classes which connect to existing or legacy data sources used by other meteorological applications. This is of particular interest to all NinJo partners, because they are able to connect to back-end systems outside the NinJo server/data tier or to perform comparisons between NinJo and other systems. With the aid of these adapter classes it’s also possible to use different middleware implementations. Therefore, data can be read from the local disc (flat file), through a socket stream (legacy software), or using Corba, the NinJo standard.

The ability to access local file data by means of the CAL has already proven to be invaluable during development. The client layer developer has a fast way to work with real data, while the data server is being developed. Researchers can exploit the file access feature to quickly display their data. Since, both data sources implement the same service interface, all that is needed to switch from the file system to the data server is to adapt the name of the implementation class in the configuration of the CAL.

4.5 Data Servers

A generic Meteorological Workstation needs to be able to import and store all sorts of standardized data. The NinJo data server infrastructure consists of several types. One data server hosts a set of data types with common storage and access characteristics. Examples are point data servers, gridded data servers, or product data servers. All data servers follow one generic design to facilitate the implementation of new data types.

NinJo servers import data from the file system and from a decoding component. File import allows for only a few possibilities, like gridded data as GRIB-files or certain geo-referenced image products. The decoding component, called GLOBUS, imports all observational data (ASCII or BUFR), and sends it after conversion to the import service of the point data server. NinJo partners who don’t want to use the GLOBUS decoding system, can utilize the import service directly, because it supports external data formats. All data is transformed, for performance reasons, into special internal formats. We use netCDF for gridded data, multi-resolution/multi-tiled geoTIFF for image products, and optimized byte arrays for point data objects.

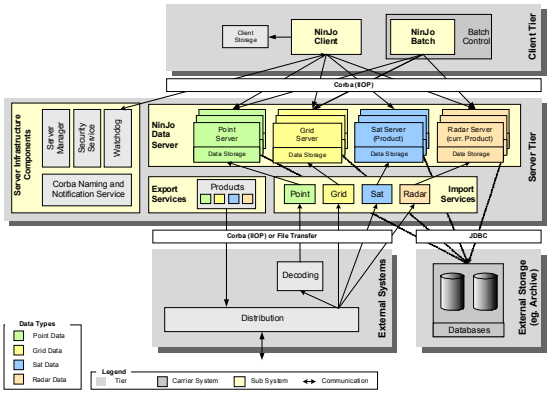

Figure 3: The NinJo tiers. Shown are the most important servers with their import/decoding components

In order to make the server as generic as possible, the design makes no assumptions on the middleware. Servers, like all other NinJo components, are configurable. Server configuration will be supported by graphical tools

5. THE MANAGEMENT OF THE NINJO PROJECT

The success of an international workstation project not only depends on the technical design but also on the management of the whole process. Very few team members had experiences from previous international software development projects. So we were facing a real challenge. Currently, NinJo is a project consisting of 5 institutes and 2 consulting companies. Development takes place at 8 sites., with roughly 50 people involved. This team has to be “developed” and “nutured””, the flow of information has to established, and the development of the software itself has to be formalized.

5.1 The Project Organization

With so many diverse partners and requirements, good software development practices - requirements gathering, documentation, user testing, project planning, estimation, scheduling, reviews, evaluations - have to be in place. They can not be ignored. Requirements are gathered amongst the consortium members and the work is assigned to individual teams. This allows access to a much larger pool of expertise and a much better

critical review of the software. A big advantage of working within an international consortium is that petty issues that can bog down a project disappear and national pride provides incredible motivation. There is very strong commitment and support at all managerial levels within each organization for the project – often a key success factor.

The most important organizational body is the steering committee. It is responsible for the assignment of resources and the prioritization of tasks. Every partner appoints one member to protect the partners interests. The project office is located in Offenbach

Although a lot of our Danish partners understand German, we have switched the working language from German to English after DMI joined the project. This was "very much appreciated" by the MSC. The addition of MSC created additional issues – distance, travel and time differences. With telephone, email and telecommunication networks, the distance and time is not such a hindrance. Such a substantial project would not have been contemplated 10 years ago before the advent of email and the internet. In fact, the 6+ hours of time difference promotes better planning and better communication as one tries to prevent emergencies.

Very early in the project it became clear that we had to minimize the amount of communication between the different sites. Therefore we concentrated work packages to single sites. The Berlin site is responsible for the PAC framework, the radar layer is handled in Toronto. This is analogous to good software practices – interfaces are clear, thin and well defined. There is only one team that consists of members from all partners - the Design team. They meet regularly to discuss software design issues in order to be able to integrate all requests from the partners. There is one chief designer to make sure that there are no diverging design concepts.

Our design principle, the separation of basic infrastructure components (framework) and specific applications (layers), makes software development "across the Atlantic" possible. Even though the amount of communication is reduced during the development of a specific application or layer, it still is very intense during the specification phase.

5.2 The Integration of the User

Since objective performance figures, like reduced lead times, are not available at this stage of the project, the forecaster is the ultimate judge of NinJo's success.

We have integrated the forecasters from the very beginning. They helped formulate the requirement specifications and are integrated in the refinement phase of the respective work packages when we deal with GUI-components.

The most important NinJo project body is the evaluation group, where forecasters from every partner get together to test releases. The fruitful exchanges of ideas within this group helps to enhance the functionality. Through the participation of forecasters from each partner, the view on how to present NinJo to the forecaster gets broadened. In the last review, forecasters indicated the need to have quick access to their use cases without clicking too much - the GUI was too cumbersome. This lead to a window containing application (layer) specific panels with shortcut buttons. A satellite panel offers for example: visible, infrared, water vapor single channel, and multi channel images. Now the most important use cases are only on click away.

Through the evaluation team, forecasters recognize that they are an important part of the development process and that their contributions will make NinJo an even better tool.

5.3 *Making the Project Work Technically*

The most important tool that's needed for developing software in distributed teams is a configuration (CM) management tool. CM tools keep a history of the changes, allow to share work and source code easily. We have chosen a commercial product, called Perforce, which is client server based. Perforce is very fast, since it's metadata driven. We have established a central repository in Offenbach (DWD's central office) that is accessed easily by any team member. Perforce not only stores sources but also the project documents. A working, easily accessible repository with a reasonable depot structure is one of the prerequisites of successful software development.

The choice of IDEs is no longer an issue, there are a many good ones around: Eclipse, IntelliJ, or Netbeans. TogetherJ is used to develop the design and to do formalized code testing. With the help of TogetherJ, one can keep source code and UML-diagrams in agreement with each other. Bugzilla is used to track bugs and to enter enhancement requests. Bugzilla is also implemented in Offenbach. Each team member has access.

6. STATUS OF THE NINJO PROJECT

NinJo 1.0 will be introduced operationally late 2004/early 2005. NinJo 0.7, which is the most recently evaluated version, features most of the required data types and servers:

- Interactive multi window, multi screen display
- Batch
- Point data (FM12/13/15)
- Gridded data
- Geo-vector and geo-raster data

- Satellite imagery (geostationary satellites only)
- Radar imagery (image products only)
- Cross Sections (Gridded data only)
- Stream lines (Gridded data only)
- Diagrams
- 3D-Visualization (Gridded data only)

There will be two more releases before the 1.0 release. One of these will be evaluated by forecasters. Short term activities include:

- interactive chart production,
- data modification functionality,
- integration of MSC's radar-software (CARDS),
- the addition of several new layers (e.g. lightning) and associated data servers,
- alerting and monitoring

7. REFERENCES

Greaves, Brian and Robert Paterson, Ros Trafford, Norbert Driedger, Paul Joe, Norman Donaldson, 1999: Development of Shorter Range Forecasting Tools in the Forecast Production Assistant (FPA). Fifteenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology, Dallas, Texas.

Joe, P., Marie Falla, Paul Van Rijn, Lambros Stamadianos, Trevor Falla,
Dan Magosse, Lavinia Ing and James Dobson, 2003: Radar Data Processing for Severe Weather in the National Radar Project of Canada, 21st Conf. SELS, San Antonio, TX, AMS, 221-224.

Kusch W., H.-J. Koppert, and M. Pogoda, 1994: The Meteorological Application and Presentation System (MAP) of Deutscher Wetterdienst (DWD), *Tenth International Conference on Interactive Information an Processing Systems for Meteorology, Oceanography, and Hydrology, Nashville,* Amer. Meteor. Soc.,200-203

Koppert, H.-J., Schröder, F, 1997: 3D-Visualization at DWD, *Sixth Workshop on Operational Meteorological Systems, Reading, UK, ECMWF Workshop Proceedings*

Koppert, H.-J., Haase, H., Gehrke, O., Lehmann, S., 1999: Lessons Learned from Developing a Prototype Meteorological Workstation with Java, *Seventh Workshop on Operational Meteorological Systems, Reading, UK, ECMWF Workshop Proceedings*

Lapczak, S., E. Aldcroft, M. Stanley-Jones, J. Scott, P. Joe, P. Van Rijn, M. Falla, A. Gagne, P. Ford, K. Reynolds and D. Hudak, 1999: The Canadian National Radar Project, *29th Conf. Radar Met., Montreal, AMS,* 327-330.

Verret R., D. Vigneux, J. Marcoux, R. Parent, F. Petrucci, C. Landry, L. Pelletier and G. Hardy, 1999: Automation for an improved efficiency. Prerprints15th International Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, AMS, Dallas, Texas, Janauray 10-15 1999, 72-75.