UDC 004.9:504:519.6

FIRST PROTOTYPE OF REDESIGN OF THE RODOS SYSTEM FOR NUCLEAR EMERGENCY MANAGEMENT IN EUROPE

I. Ievdin, O. Prymachenko, N. Shlyakhtun, I. Kovalets, D. Treebushny, G. Donchyts, I. Chorny, O. Guziy, M.I. Zheleznyak
Institute of Mathematical Machine and System Problems NAS of Ukraine
e-mail: yewgen@env.com.ua

INTRODUCTION

The RODOS system for nuclear emergency management in Europe had been developed under the funding of European Commission from 1992 (Ehrhardt, et.al., 1997). RODOS had been designed as a complex system, which includes models and data bases for modeling and evaluation of all the consequences of nuclear emergencies and for the emergency planning. Originally RODOS system was developed as HP UNIX software. Recently the work had been started concerning the redesign of the RODOS system, which addressed the following main topics: 1) developing multi-platform system with ability to run on PC; 2) developing distributed system; 3) databases redesign; 4) modern GIS and UI; 5) easy administration of the system with data base functionalities; 6) easy integration of new simulation models. In the following sections the main technical solutions are described and the key software functionalities are presented.

TECHNICAL SOLUTIONS

The overall structure of the RODOS system is presented at the Figure 1. As it is seen from the figure the main system parts are: Models, User Interface (UI), GIS, Data Bases (DB), and system Kernel. As it is seen from the Figure 1 all system parts communicate with each other only via system Kernel.

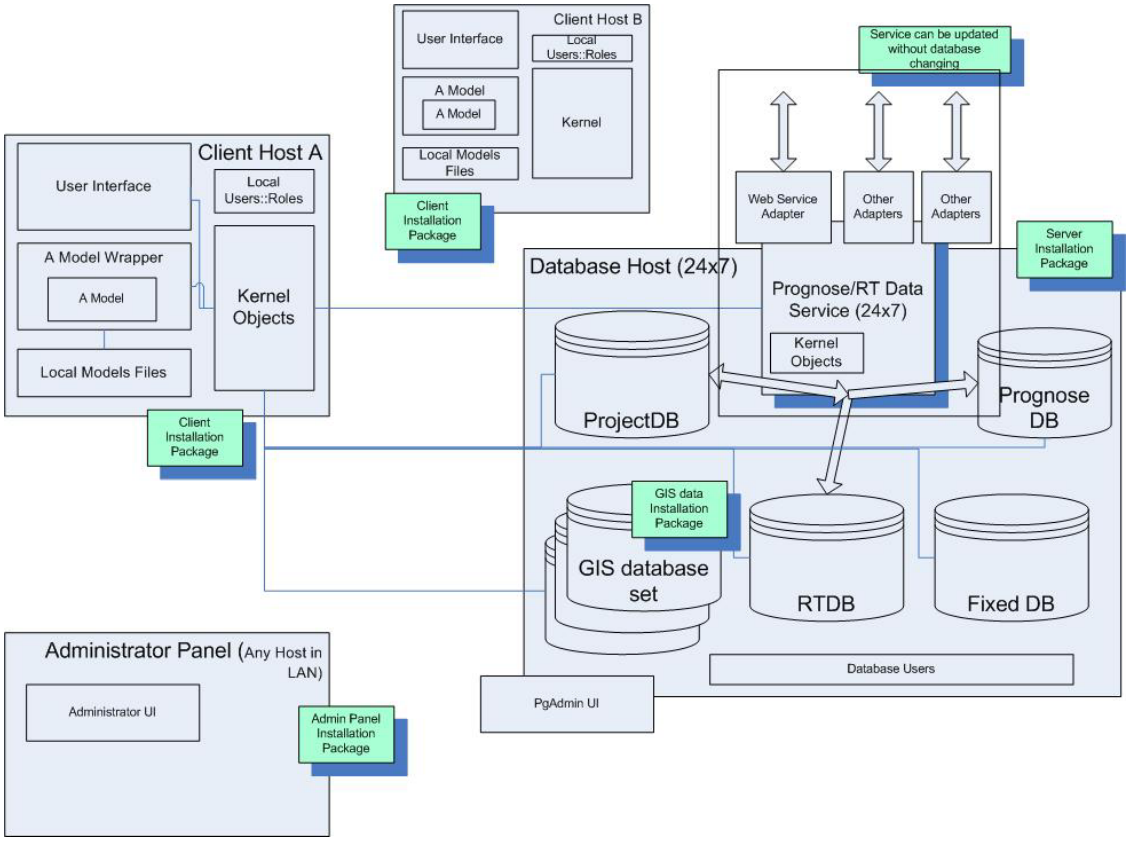

Figure 1. Overall structure of the RODOS system

RODOS are planned as distributed system with main host and several client hosts. The main host contains databases running and set of services. Those services are used for access RODOS system from other systems (incl. other RODOS instances), accepting income data, for example from numerical weather forecast providers, working with RODOS system via web interface and broadcasting RODOS data for subscribers. The models are running in the client hosts, which also contain user interface, main part of kernel objects.

System kernel

RODOS system can be viewed as two-level software: there are the level of design and level of realization. Most of design-level entities are interfaces and abstract classes (in java programming language) while they are implemented and extended in realization level. Level of design organizes RODOS framework. The framework has two major and very important goals: 1) to keep system design and its implementation separated and 2) to allow easy extensibility via plugin concept. The main framework characteristics can be enumarated: 1) user interface has to be structured; 2) data are organized hierarchically; 3) the framework has to provide infrastructure for models managements incapsulated in projects, models input/output displaying, including GIS controls, working with SQL databases, multiuser, distributed environment with main host, web functionality (both API and UI).

Domain specific data inside of the system (kernel objects in Fig. 1) is organized in the form of dataitems. This words is simplification of the sentence „Hierarchy of IDataItem successors". IDataItem interface was firstly introduced in Donchyts and Zheleznyak, 2003 and was further extended during the work on RODOS redesign. DataItems classes are organized as *Composite* pattern (see 4.2), this means they has hierarchic structure and could contain large hierarchical structured data sets of different types which a model may need. They allow set of functions for support of the most important manipulations with DataItem objects. Those manipulations could be categorized in following main types: a) manipulations for support of handling data semantics, such as set and return name, description, datarole, quantity and substance of the dataitem object; b) manipulations, for handling of the hierarchical data structure, such as: for given name and index returns dataitem that is direct child of this dataitem; set and returns parent of this dataitem object; add, remove and return list of dataitems that are direct children of this dataitem; c) defines the entry point for visitor object which provides responses for the external events (e.g. mouse clicks).

Databases

The following databases were designed and created (see Fig. 1): ProjectDB database contains models runs with messages, input and output data stored either by reference or by value; PrognoseDB database with numerical weather forecast data; Real time data base (RTDB) with measurement data; FixedDB database with fixed (e.g., site location) data. All databases currently are managed by the same PostgreSQL instance.

Model wrapper

Model is typically Fortran code. Connection of models with other parts of the system is performed through model wrapper (Fig. 1). Model wrapper is the core entity in the model integration. It is the only entity in the system that is designed "to see" both system kernel API and model top level functions (MTF). For a particular model, model wrapper is the class that implements **IModelWrapper** interface in the system framework. The interface is as general as possible and represents the general logic of model usage. We recognize 5 methods: for model initializing, for running the main cycle, for retrieving results from the model to system kernel (results will be further propagated to databases, user interface etc), for setting and getting library name. First three methods accept parameters of **IDataItem** type. Model wrapper is designed on the base of **IModelWrapper** interface and implemented by model integrator. In the present version of the redesigned RODOS system the atmospheric dispersion model ALSMC had been implemented via model wrapper.

GIS

In RODOS software GIS subsystem includes: GIS controls with tools (GIS UI component), GIS data storage. GIS controls and tools are built on top of the Geotools 2 library, which is one of the most widely used open-source GIS library for the Java platform. GIS controls present geographical and georeferenced data to the user in the form of maps, and allows users to manipulate these data. GIS UI component has three distinct components – **the map control, the layer list, layer and symbolizer editors**. User interacts directly with these three components; all of them use geotools library extensively. The **map control** is the component which actually displays the map and the geo-referenced data in map form. It allows the user to navigate the map, zoom in/out, measure distance and area, and select, print and export to image geographical features presented on the map. The **layer list** permits the user to manipulate and customize the map: adding new layers, moving, changing and hiding layers, import and export layer data to external files in standard formats.

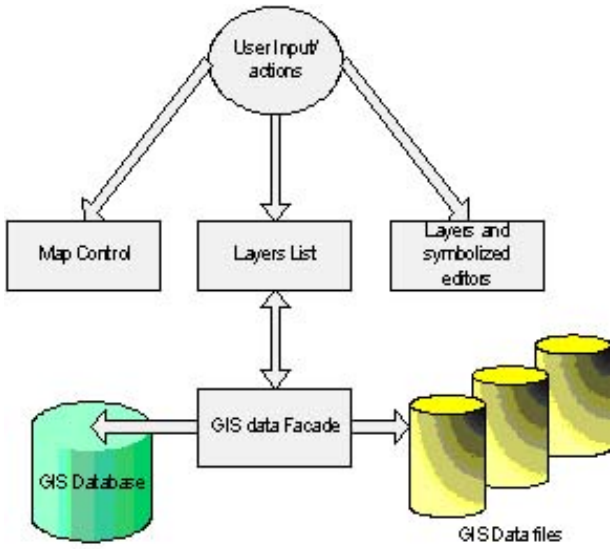

Figure 2. GIS UI and GIS data storage

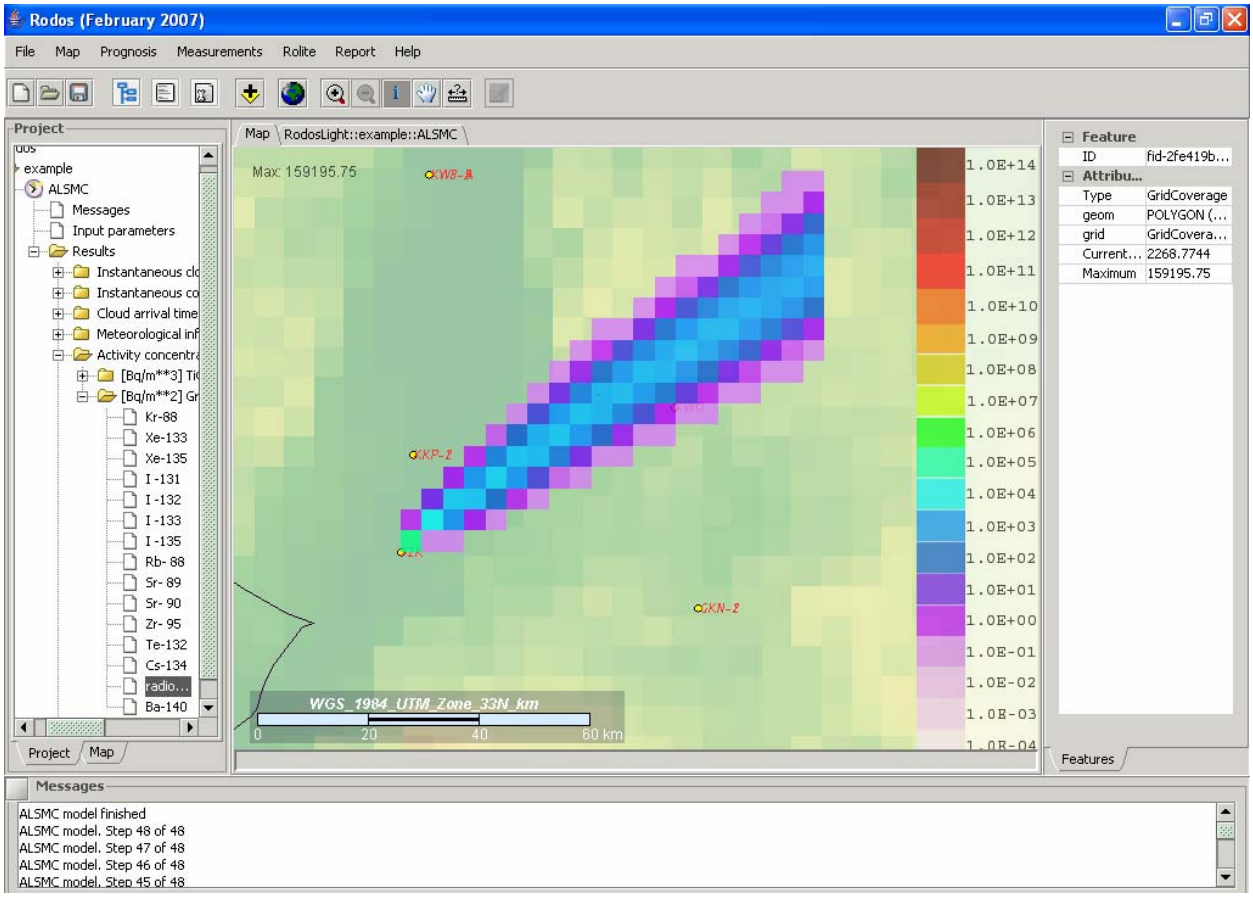

Figure 3. Main window of redesigned with the different subwindows. Map window contains results of calculations of Cs-137 deposition following the release from FZK nuclear power plant

The **layer and symbolizer editors** allow the user to change the list of symbolizers associated with the layer, edit particular symbolizers, and import and export symbolizers to external .sld files. The GIS related part of Data Façade (**GIS Data Facade**) component provides utility services to the three other components, interfaces between RODOS data representation model and the Geotools data representation, and allows the system to display time-dependent data in a simple and intuitive manner (as a slideshow or as individual time frames). This component also performs geometric operations: it creates contour plots from grid data, calculates distance and area. The input data for GIS subsystem falls into three categories: user input, external data files and RODOS data (from database or from model runs). GIS output usually is visualized, however geo-referenced data may also be exported to external files, and map images may be printed or exported in a variety of formats.

Organizing projects.

Model runs are organized in projects. Tasks are entities that hold models metadata and all information related to run: input, output data, messages, warnings and errors that ocurred during model run. Application object is a kind of singleton that holds opened projects. The whole structure is visible (reflected in UI control) in **ProjectExplorer** Window of the user interface (see below) at runtime. Project lifecycle includes such activities: creating, working with tasks, storing in database, reusing as template to build other project, reviewing and comparison with other projects. Project can be deleted from database by system administrator (superuser).

User Interface

The desktop user interface (UI) is organized mainly as desktop application. Web user interface that is also planned plays secondary role. The UI has several main parts. The main window of RODOS is shown at the Figure 3, which consists of main menu, toolbar, project explorer/map explorer window, message window, property/report window, input window (hidden at the Fig. 3), and map window, where results of calculations can be displayed (Fig. 4). The UI upper level interfaces are part of the RODOS framework. Developers will implements those interfaces while writing custom UI controls – UI plugins.

CONCLUSIONS

The first prototype of the redesigned RODOS recently was designed. The following activities were performed: Main data types were defined and realized in the system kernel; all main databases were designed and created (Fixed DB, Real time DB, Prognosis DB, Project DB, set of databases for GIS data); User Interface (GIS window, input Rolite windows, Project Explorer, Properties Panel, Message Box, menu, toolbar) was implemented; The atmospheric dispersion model (ALSMC) was particularly integrated; GIS control with common ESRI Shape format and raster data (GeoTIFF) implemented; Report framework designed and implemented; Measurements and Numerical Weather Forecast Data can be loaded into databases and some data can be displayed in form of grids in GIS control and charts; Installation procedure were defined and installer implemented.

Recently first releases of redesigned RODOS were distributed among the RODOS users group for preliminary evaluation. A number of comments received indicate generally good opinion about new RODOS, but a lot of future work is foreseen to achieve full functionality of the old RODOS and to further extend it.

References

1. Donchyts G., Zheleznyak M., 2003. Object-Oriented Framework for Modelling of Pollutant Transport in River Network. In Computational Science – ICCS 2003, Lecture Notes in Computer Science, Springer, Vol. 2657/2003, 0302-9743.

2. Ehrhardt J., Brown J., S. French, G.N. Kelly, Mikkelsen T. and Müller H.: RODOS:Decision-making support for off-site emergency management after nuclear accidents. Kerntechnik 62 (1997) 122-128.