



## **Information systems for simulation of train operation**

L.A. van Bokhoven, J.A.H.M. van Keulen  
*Holland Railconsult, Postbus 2855,  
NL 3500 GW Utrecht, The Netherlands*

### **Abstract**

Netherlands Railways has developed several simulators, each of which has a separate aim, interface - often deficient and not very efficient - and storage format. Now it has decided to develop a new graphical user interface, a uniform data model and future-proof system architecture so that simulators can be used more quickly, more effectively and in combination with one another and so they can be properly maintained.

This article describes the conceptual ideas, the architecture and several components such as the graphical user interface and the simulation database.

### **Introduction**

Netherlands Railways is constantly looking for new technology to increase the capacity of its heavily overburdened system within the confines of minimal space. Growing complexity makes it increasingly difficult to estimate the consequences of process and product innovation. Analytical methods are inadequate. Simulation has provided solutions on a number of occasions.

Simulation of the train service can be used at various stages of the development process. However, the design problems and accompanying questions vary at each stage as does the need for data to create a suitable model. Clearly a strategic study needs different data than a study about optimising the positioning of signals. That explains why different models have been constructed, each with its own import and export programs, data format and simulation tools.

In most cases the models use the same information, but with a different level of detail. The information the models require is already available in one form or another in files or databases of operational and development environments.

## Description of the problem

Theoretically speaking, the simulation techniques used by Netherlands Railways provide users with all the facilities they need to carry out simulation studies. However, each simulator operates in its own way, as figure a shows.

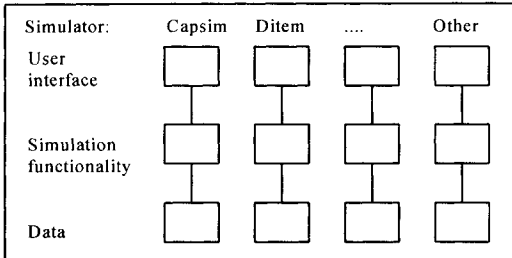


Figure A: Current situation

In the present situation, each simulator has its own user interface - often highly deficient - and stores the data in its own distinctive format. This creates the following problems:

- Entering and modifying data is time-consuming and error prone.
- Users must be familiar with numerous details of data storage formats.
- Import and export data are not easy to re-use.
- Users have to master several interfaces.

NS uses several different simulation techniques which were originally developed for a specific purpose: route section, junctions or traction energy. Depending on the stage of development and the problem which needs investigating, there is a need for one or more simulators. This creates an additional problem:

- Simulations carried out as part of one project with one or more simulators create twice or three times as much work.

This situation is annoying for everyone, developers included, especially bearing in mind that the simulators were compiled using different programming languages and development environments:

- System modules with comparable functionality cannot be exchanged with one another in any way whatsoever
- Developers need to be familiar with all the details of all data storage formats.
- Developers have to know several programming languages and be familiar with different development environments.
- Importing data from NS sources has to be programmed several times.

## Solution

Using and maintaining the existing simulators took too long and cost too much money. Something had to be done. The idea was therefore put forward of developing a uniform graphical user interface, an unambiguous data model,

flexible future-proof system architecture and a pragmatic technical approach. It will solve the problems that have been described because:

- A graphical user interface produces much higher productivity.
- The same interface is always used.
- Users no longer need to know the data model.
- Data only have to be imported once.
- Importing existing data for each source only has to be programmed once.

Integration of the user interface and data storage will eventually produce a "family" of simulators and the non-relevant differences between simulators will disappear.

### Graphical user interface

The concept illustrated in figure b has been developed to resolve the problems. As it shows, in future all simulators will use GUIS, the Graphical User Interface for Simulation, and data will be stored in a database using the same data model.

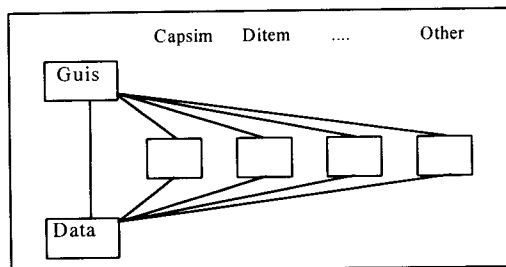


Figure B: Future situation

Introduction of a new graphical user interface means that the graphical interfaces of the existing simulators will no longer be used. That is not a problem because the real value of existing simulation techniques lies in the simulator itself.

### Model architecture

The user interface and the data model are based on the concept shown in figure c thereby meeting the need for different abstraction levels within the same system.

At present, there are two abstraction levels, network level and infrastructure level. The network level is made up of primary areas (control points and open tracks). Primary areas are elaborated in more detail at the infrastructure level consisting of one or more layers with varying degrees of detail.

**Network level** The network level contains an overall description of each primary area. For a station, for example, the data would include the number of incoming and outgoing tracks and the number of platform lines, for an open track whether it consists of one, two or four tracks. A rough simulation can be made using this general information.

**Infrastructure level** The infrastructure level describes the necessary details of a primary area. Structure is introduced by splitting the information into layers. Each layer describes a specific aspect; for example an iron layer is where the topology is recorded, a signal layer is for the safety system.

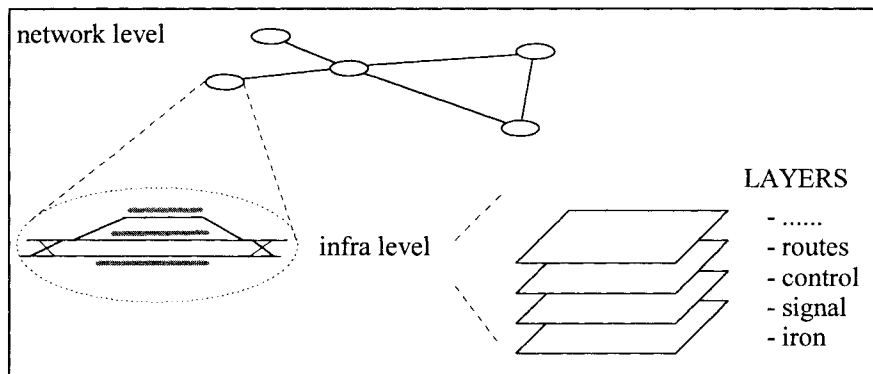


Figure C: Abstraction levels

**Variants** The detailed structure which is created with abstraction levels and the layering mechanism serves as the basis for the flexible definition of different model variants within a project. For example a primary area can be defined in two different ways as P1a and as P1b. One model variant is currently defined as P1a, P2 and P3; the other as P1b, P2 and P3. The new system offers enormous flexibility which can be used to define different model variants and compare them with one another without duplicating work, such as more or fewer platform lines in functional terms or a signal or radio block safety system in technical terms.

### System architecture

New system architecture has been developed so the new user interface and database can be combined with the existing simulators in a way that is flexible and future-proof. The new system architecture comprises sub-systems which are independent of one another as far as possible. This means they can be run together as a single application on a computer and can be installed on different computers linked in a network. The big advantage of this setup is that the overall system can grow with the requirements of users but there is no need for users to give up the familiar PC interface. Additionally, the database and simulator can easily be installed on one or more powerful network servers. Good architecture guarantees that future developments will be able to build on elements that are currently being developed.

### Hardware and software architecture

The platform for the graphical user interface is PC hardware with Microsoft Windows NT or Windows 95 as the operating system. This combination offers

the best guarantee that the new development will become available to a large group of users without excessive investment. The development environment will use the same platform with Microsoft Visual C and Microsoft Access in combination with OO-design tools.

A powerful Pentium machine with at least a 17" screen, 32 Mb RAM and a 540 Mb hard disk is enough to run all sub-systems on one computer.

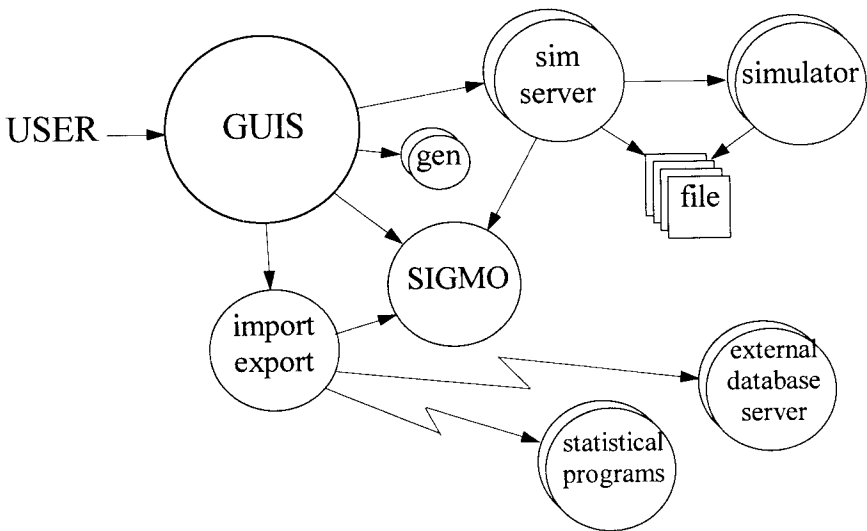


Figure D: System architecture

## System components

The system architecture in figure d shows the internal structure of the system architecture and the relationship between the various sub-systems which are described below.

### GUI

GUI is the working environment for users of simulation techniques. With the help of the graphical user interface users design the simulation model, run simulations and analyse the results. One of the most important functions of the user interface is to provide users with optimum support so they can work more effectively and efficiently.

GUI is based on a Multiple Document Interface (MDI), which means that it uses an application window and other document windows which can be opened at the same time such as:

## 352 Computers in Railways

- Project document, a meta-file with references to the other documents.
- Network document, with an overview of the primary areas.
- Infrastructure document with a detailed overview of the infrastructure of a primary area.
- Timetable document, with an overview of trains and train series.
- Results document, with data relating to a simulation.

GUIs has a menu structure and seeks to achieve maximum consistency between different document windows. Depending on the document, users can perform different actions, examples of which are given below.

**Open simulation project** The user selects an existing simulation project or creates a new one. Both the project document and the network document are opened.

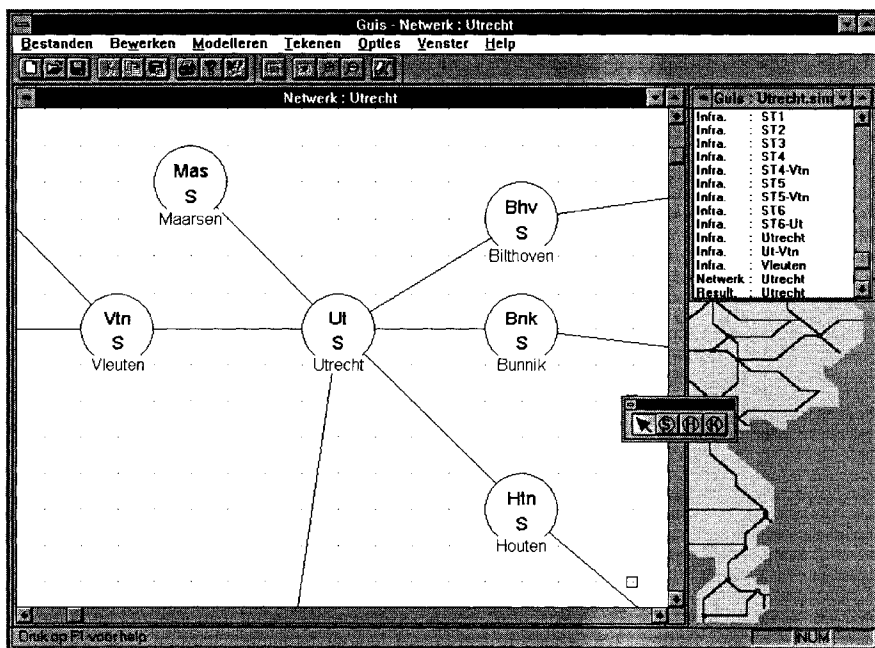


Figure E: Example network document

**Model network** A document manages the data. A view mode shows the data and provides the interaction with the user. A floating tool palette supplies the objects that can be placed. As an optional feature, a specific cursor can be created so the user receives visual feedback. Modelling (drawing) is supported by a grid. Users can select, move, cut, copy and paste, and delete objects. Right-clicking activates a context-sensitive menu allowing the user at least to open a object property dialog box. Much functionality needed for network modelling is

common to topology, safety and energy supply modelling. Double-clicking on a primary area in the network document opens the infrastructure document for the primary area.

**Model topology** This is similar to network modelling. Editing features such as rotating and inverting must be implemented. Right-clicking activates a context-sensitive menu offering at least a property item. TopologyView shows points, buffer stops and route sections.

**Model safety** The topology is always visible in the background. The user places safety objects such as joints, signals and signs along route sections and links them to the route sections. The objects are available through a floating tool palette. Editing features such as rotating and inverting must be implemented. Right-clicking activates a context-sensitive offering at least a property item.

**Determine trains** Two dialog boxes are needed here as well, one to consult existing trains and the other to change and add trains. Determining a train takes place by completing a table. The table for a train consists of rows and for each control point a number of columns such as arrival time, track, through time, track, through time, track and exit track.

**Consult distance-time curve** Selection of one of the simulation results. A choice of the route (part of a route) and a time window. The curve settings can be changed in a dialog box.

**Other actions** Many more actions are defined such as Determine route, Model energy supply, Determine rolling stock, Determine train sets, Determine simulation model, Set simulation, Run (test) simulation, Show animation, Consult track occupancy, Consult punctuality, and many more....

## Sigmo

The Sigmo database (Simulatie Gegevens MOdel = Simulation Data Model) stores the data in a uniform way. The database enables links to be established - without too much additional effort - with external systems such as existing databases where infrastructure data are stored or with a statistical program so the results of a simulation can be analysed. The application is easily scalable because the interface to Sigmo is based on ODBC and the data model is described in SQL. Sigmo can be implemented with both Microsoft Access and with Oracle.

## Simserver

The simulation server is the interface between the existing simulator on the one hand and the user interface and the database on the other. The simulator interface depends on the simulator to a large extent, but the interface with the user interface

## 354 Computers in Railways

and the database can be virtually the same. Making the identical parts available in the form of a collection of procedures (DLLs), enables a specific interface to be implemented fairly independently of the system. Existing NS simulators can be linked by the internal team, and other development teams will be able to link up their own simulators in the future.

### Simulator

The simulator receives its information from the accompanying simserver. It executes the commands it receives and generates results on the basis of the model which has been entered and the accompanying parameters, which can be presented by GUIs and/or (partly) stored in the database.

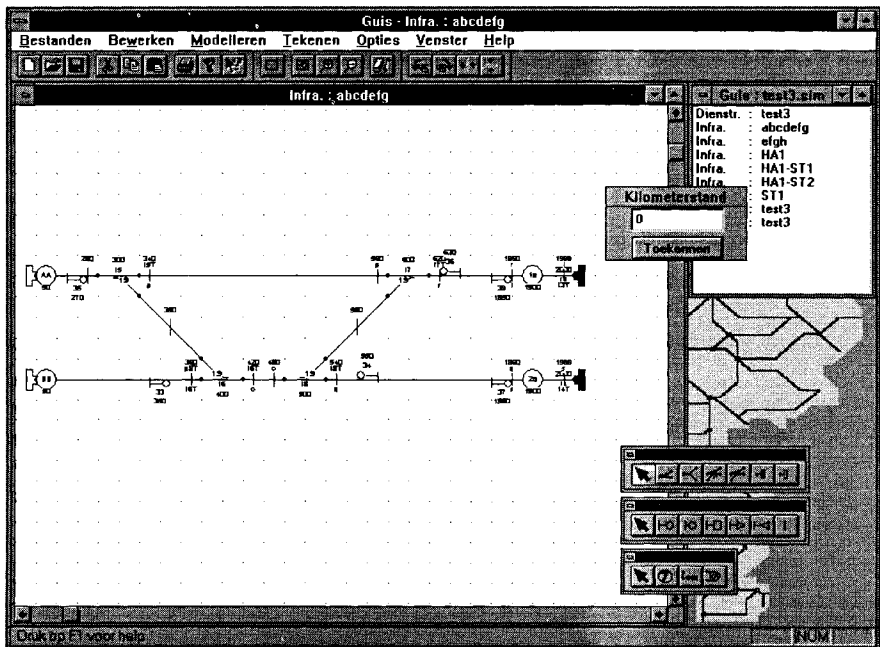


Figure F: Example infra document

### Generators (gen)

A number of generators can be included to provide the user with additional support. They generate new design data on the basis of limited input in combination with a number of standard design rules. In most cases this will be sufficient. However, users can make manual changes in order to optimise the model if they wish. Several examples of generators are listed below.

**Joint generator** The joint generator generates joints around points and behind signals by examining the suitability of every piece of track for joints. Route



sections which already have joints are not included because the program assumes that the user has already dealt with the problem for this particular section. In such cases information is not overwritten.

**Track generator** The track generator examines the topology of a control point and generates a list of all possible routes which a train could take, such as from the arrival track to a platform line. When designing the timetable the user can select routes and set a specific priority.

**Route generator** The route generator examines the topology at the highest abstraction level and generates a list of all possible routes. The user can delete routes that are irrelevant to the timetable in question.

**Signal generator** Other examples of generators include a signal generator or even a control point generator. More of these generators can be and will be developed in the future.

### **Import/export facility**

NS has several databases containing information which can be important for simulation purposes such as infrastructure data, timetable data, rolling stock data and delay data. Obviously it is sensible to use these databases instead of re-entering data by hand every time.

It may also be desirable to run the results of a simulation through statistical software for further analysis. In both cases there is a need for a link, which can be created with the help of a disk or through a network with on-line access to databases.

As with the simserver, the facility is based on a DLL which includes the generic components. It can be used to write new import and export facilities.

## **Conclusion**

Much of what we have described in this article has already been developed by now (March '96). The final touches are being added to GUI version 1.0 and a pre-release is currently being tested. It has an interface with one of the simulators and there are plans for a link with a second simulator. The first version would seem to be a good basis for further development.

Further development is desirable because - as so often happens with new developments - it has been impossible to meet many of the legitimate requirements within the available budget and lead time. We are also convinced that although development has been closely coordinated with future users, the proof of the pudding is in the eating. Once users actually work with the system we expect them to come up with lots of useful suggestions for improvements.