A new Graphical User Interface for CAPRI

Wolfgang Britz
Institute for food and resource economics
University of Bonn

- preliminary version, do not cite -

Bonn, June 2006

Acknowledgments

Many people have over the years contributed to the development, maintenance and application of the CAPRI modelling system, and after now ten years since a first prototype was constructed, it is almost impossible to list them all and name their specific contributions. The author opted for this rather technical paper to refrain from citing the different (working) papers which shed more light on methodological questions, but rather refers in general to the CAPRI documentation.

Nevertheless, it is only fair to mention Hans-Josef Greuel and Andrea Zintl, who both long before CAPRI was born have already developed software concepts and code which underlined to a large extent until now the DBMS of CAPRI, and in parts, its Graphical User Interface. Both continue to support the technical development of CAPRI.

Finally, the work described in here would have been impossible without the funds by different donors, mainly the EU Commission.

All errors in text and code remain with the author.

The author

Dr. Wolfgang Britz is a senior researcher and lecturer with the Institute for Food and Resource Economics at the University of Bonn, and has co-ordinated since several years the activities based on the CAPRI modelling system. Its responsibilities further on include the methodological concept of CAPRI and , to a larger extent, its software implementation.

Contact:

Dr. Wolfgang Britz

Institute for Food and Resource Economics, University Bonn

Nussallee 21

D-53115 Bonn

Tel.: ++49-(0)-228-732502

Wolfgang.Britz@ilr.uni-bonn.de

Content

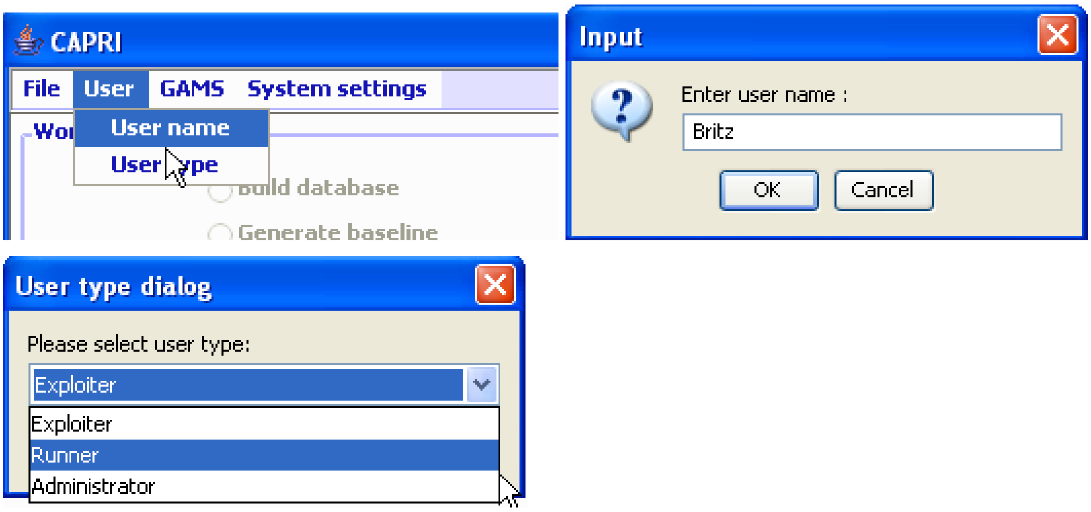
Background

The use of the CAPRI modelling system is increasing, and the user group becomes more diversified. Whereas in the first years, almost all users had directly contributed to model development and where familiar with the underlying GAMS code, more and more users now get to know about the system during training sessions, and have only a limited knowledge of GAMS and the CAPRI GAMS code. Already a few years back, a Graphical User Interface (GUI) was developed in order to supports users to apply CAPRI for simulations and exploit results. For reasons laid down further down in a short chapter, this GUI needs was now replaced by a completely new development. That paper both explains the usage of the new GUI as well the underlying software concept. It is structured as follows. The first chapter gives a short overview over the different work steps necessary to finally allow simulations with CAPRI.

Logging in

The first step is to set the user name and level, which is done by selecting the user menu from the menu bar. As long as no user name is entered, the user cannot change its type and will only have exploitation rights. The user type "runner" has additionally the right to define and run scenarios, and delete scenario results. An user of type "administrator" can perform all operations, including generation of a new data base and calibration of the modelling system.

Graph: Setting user name and user type



The different work steps

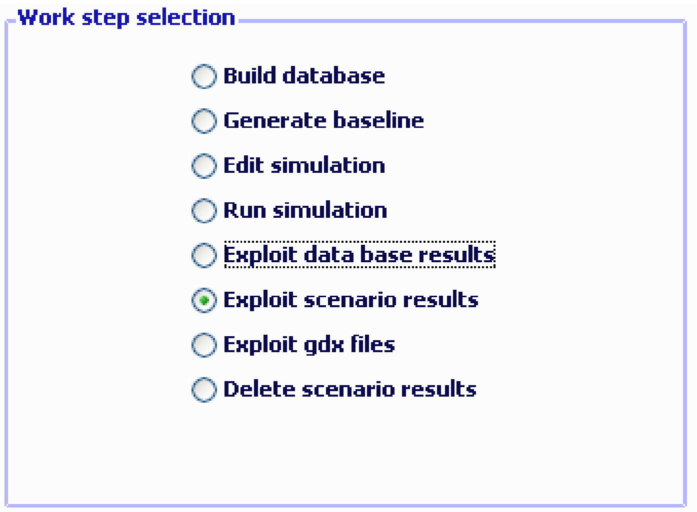
The process of using and maintaining CAPRI is split down in the following work steps:

1. Building up the data base
2. Exploitation of the data base
3. Generating the baseline
4. Scenario definition
5. Running scenarios
6. Exploitation of scenario results
7. Deleting scenario definition and results

Each work step may comprise different tasks. No task will require starting more then one GAMS program, but some tasks will start the very same GAMS program with different settings. Some tasks will not start GAMS, but other tools inside the GUI.

The different work steps are shown in a panel in the lower left corner of the GUI, and are presented by so-called radio-buttons, which means, that only one button can be selected at any time. Depending on the user level, some of the work steps and tasks may be disabled.

Graph: the work step panel



Each work step may comprise several tasks, which are shown in second panel below the work step panel. The content of the panel hence changes when the user selects a different work step. Again, the different task panels comprise radio buttons for selections purposes.

Graph: A task panel

In the following, the different tasks for each work step are described in detail.

Building up the data base

Building up the data base is the logical starting point in sequences of work steps. A new data base for the model needs to be constructed either after updates of the underlying statistical raw data, or after methodological changes in the code affecting content and structure of the data base. Controlling if the different tasks necessary when updating the model yielded satisfactory results is a time demanding task which requires in-depth knowledge about the quality of the different in-going data and the logical relations between the different elements of the data base. Users interested in ex-ante policy analysis are usually better off by taking the data base as given, and consequently, the work step is disabled for users which have no “administrator” status.

The work step consists of four different tasks:

1. *Generation of complete and consistent time series at national level, mainly based on Eurostat data (CoCO, from Complete & Consistent)*. CoCo runs per Member State simultaneously for all years, if data from other Member States are used to derive fallbacks as EU average, only the raw statistical data are used. The user can only choose which countries to run, and which years to cover.

2. *Completion of the CoCo data by time series on consumer prices and certain feeding stuffs (CONSUMER)*. In both cases, it turned out that only the complete and consistent time series for all Member States from 1. provide a good basis for that step. The step is hence run simultaneously for all Member States and years, based on the results of the CoCo task. Here, only the years to cover can be chosen by the user.

3. *Generation of time series at regional level and ex-post calibration of the model including estimation of feed input coefficients (CAPREG)*. The treatment of years in CAPREG is not identical. For all years, activity levels, output coefficients and input coefficients (excluding feed inputs) are generated. However, only for the base period, a three year weighted average around the chosen base year, feed input coefficients are estimated and the supply models are calibrated based on techniques borrowed from Positive Mathematical Programming. The user can hence choose for which Member States to run CAPREG, for which years and for which base year. Equally, the farm type module may be switched on or off.

4. *Building up the international data base*. The step includes aggregation of Supply Utilization Accounts and bilateral trade flow matrices from FAO to the product and country definitions of CAPRI, aggregation of the supply and demand elasticities from the World Food Model to the

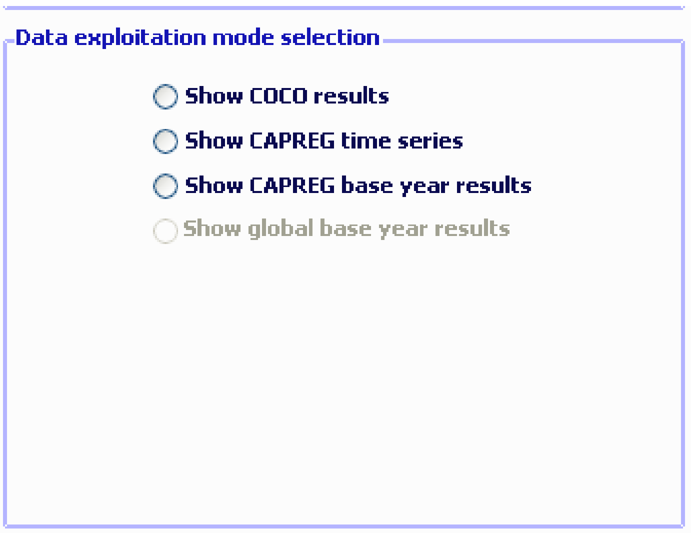
product and country, estimation of bi-lateral transport costs and conversion of the FAPRI baseline to the product and regional aggregation of CAPRI.

The underlying methodology for the different work steps is described in detail in the CAPRI model documentation. The sequence of the tasks as described above follows the work flows. It should be mentioned that certain preparatory steps, as downloading updated data from EuroStat, and converting these data into GAMS tables read by CoCo and CAPREG are no yet integrated in the GUI.

Exploitation of the data base

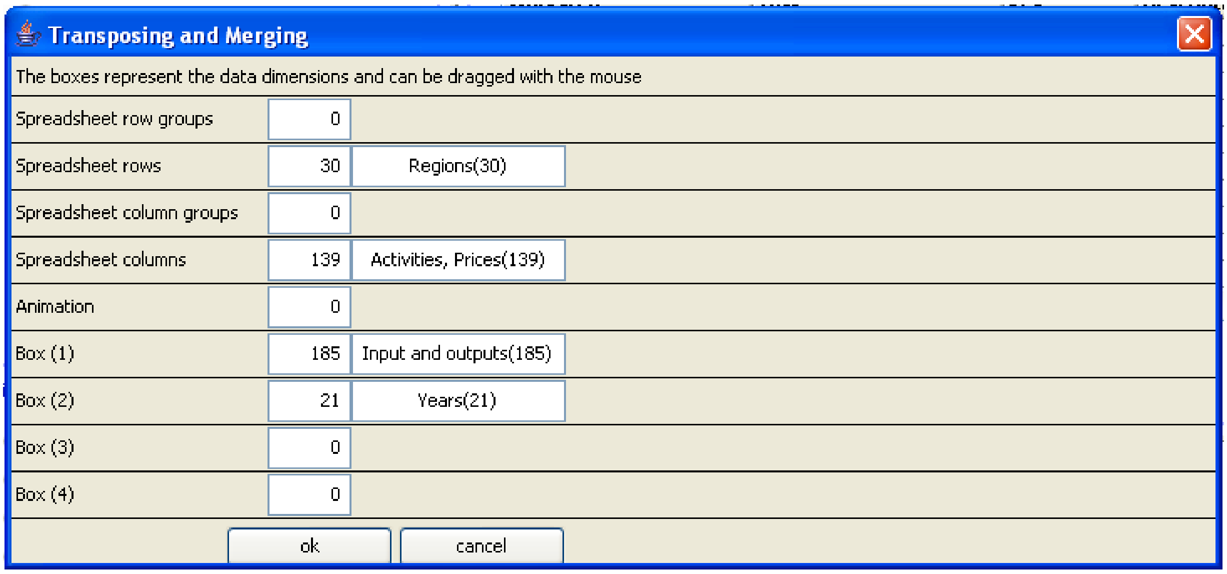
Exploitation means letting the data speak, which requires to access and view them in an appropriate way. As past requests from users inside and outside of the CAPRI network prove, the CAPRI data base with its pan-European coverage at NUTS II and now even farm type level inside NUTS II regions, along with the detailed break down to agricultural activities, inputs and outputs is valuable on its own, and not only a necessary input in the CAPRI simulation engine. The tasks attached to that exploitation steps follow the tasks described above: accessing the time series at national level generated by CoCo (market balances, activity levels and yields, unit value prices, Economic Accounts for Agriculture), time series at national and regional level added by CAPREG (input and output coefficients at regional and national level, activity levels and production at regional, income indicators) and finally, the set of data for the base period, which comprise feed input coefficients and matching animal requirements. Finally, the results from the global data base step can be accessed.

Graph: The task panel for exploitation of the data base



The results from the different steps are comprised in large “data cubes”, i.e. multi-dimensional matrices, which span the dimensions regions, columns (mainly activities, market balance items, prices, positions form the Economic Accounts), rows (mainly input and outputs, income indicators, activity levels, feed requirements) and, finally, years. The time series from CAPREG, to give an example, almost cover 200 Mio zero and non-zero data cells. When one of radio-buttons is operated, the complete set is first loaded, which may take several then seconds, depending on the processing and input speed. Afterwards, the user can determine how the data dimension should be pivoted (e.g. as time or regional series).

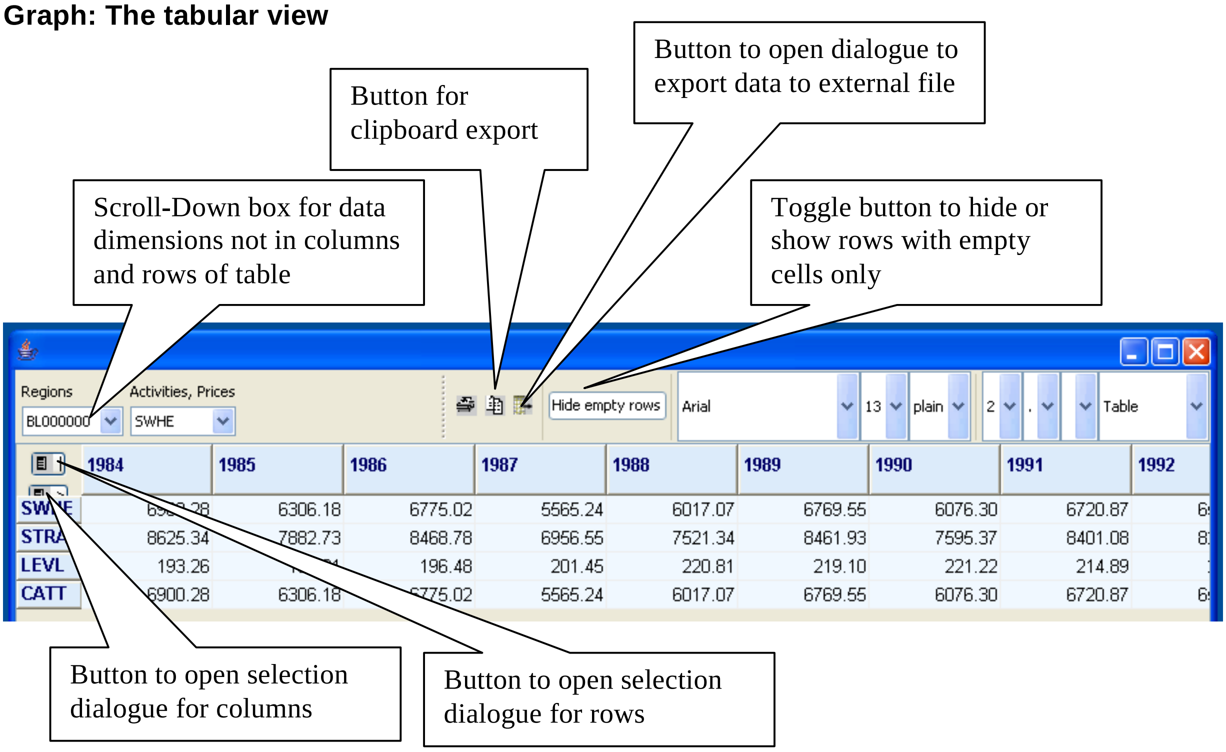
Graph: The pivoting dialogue



The pivoting dialogue is based on drag-and-drop of boxes, where each box represents a data dimension as e.g. regions or years. The rows of in the drag area are linked to the dimensions of the view port. Dragging e.g. the box for the years to the line whose label shows “spreadsheet columns” will generate a time series view. That is done by clicking with mouse in the box, keeping the mouse button pressed, and then moving the mouse. During the operation the box changes colour. By releasing the button box, the box jumps to its new position and its colour changes back to the original one. Several boxes can be dragged into one view port dimension. Views with empty rows or columns are not allowed.

To generate nicer views, the user can generate column and row groups as well. The pivoting – and as explained later, row and column selection – are maintained when exporting the data, so that the user has a quite powerful instrument to arrange data for any further processing.

Graph: The tabular view

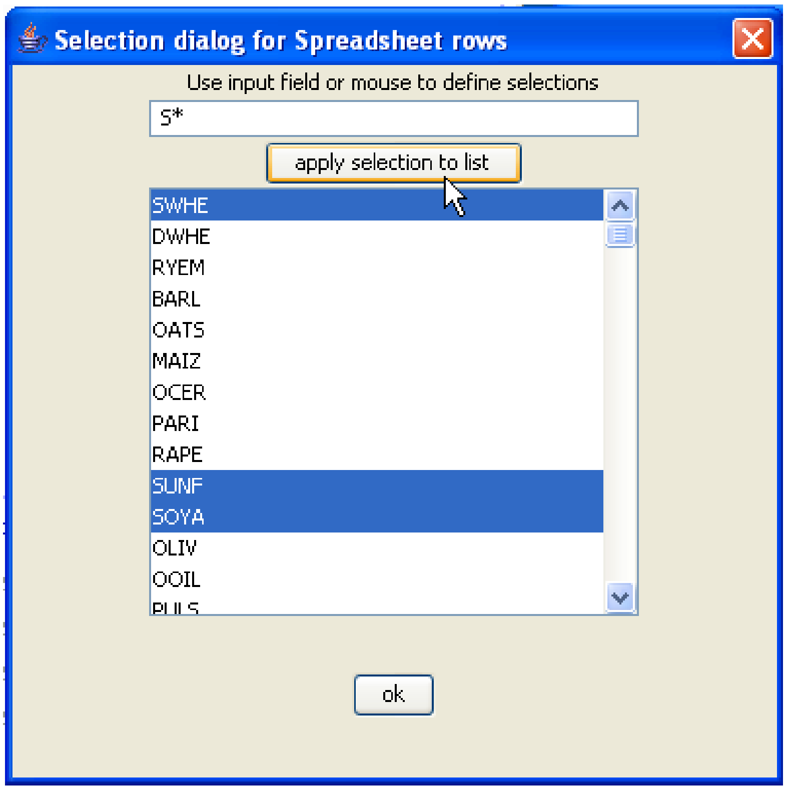


Once the pivoting done, the data are shown in a tabular view similar to MS EXCEL or MS ACCESS. The tabular view has can be controlled by the user in several ways:

- On top of the table are most probably scroll-down boxes, which carry the items of such data dimensions which are not shown in the columns and rows. With the help of these boxes, the users can rotate through the different tables generated by its pivoting. The user may however opt for a pivoting where no outer view port dimension are present, and all data dimensions are merged into the columns and rows of the tabular view, in which case no scroll-down boxes will be present.
- To the right hand side of these boxes is a toolbar which offers several options: the user may change the font family, font size and font type, the number of decimals shown, the decimal separator, and, finally, a scroll-down box to switch between the tabular view and different type of graphs. Additionally, there are several buttons, one to export the current cell selection to the clipboard, one which opens an export dialogue and a toggle button to hide or show rows which comprise zero values, only. The tool bar may be dragged away from its position, and can put above or below the scroll-down boxes.

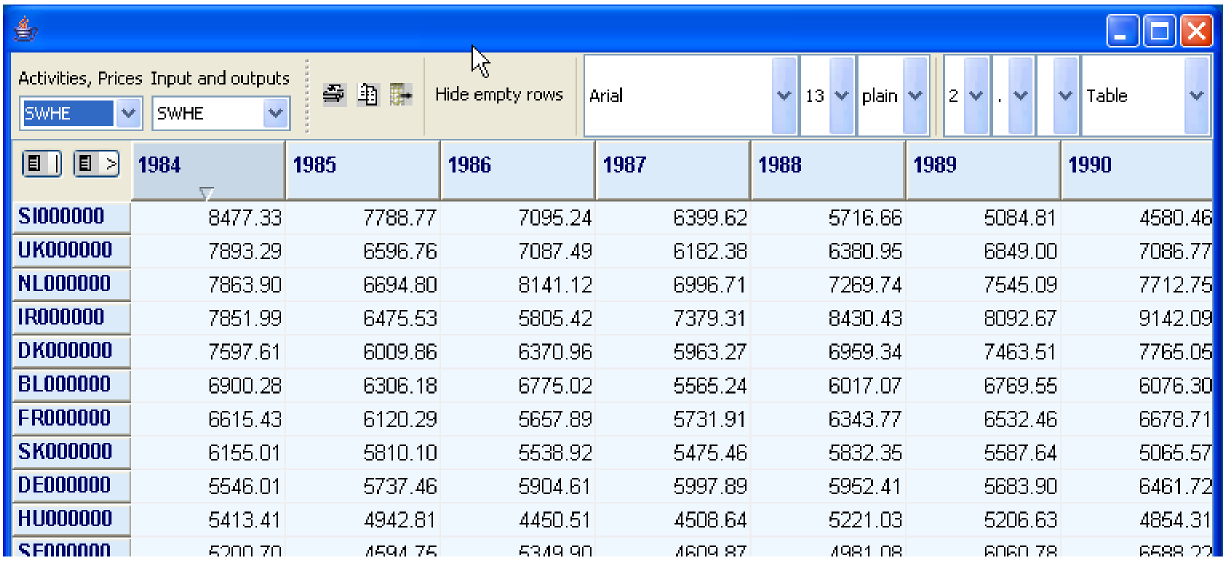
- In the corners on top of the rows label and left of the columns labels are two selection bottoms. Opening these buttons will allow the user to determine which rows and columns are currently shown, by either using the mouse or keyboard on the scroll down box, or by typing a selection string in the input field and using the "apply selection button". The selection string may either use a combination of alphanumerical chars with "?" and "*" and/or first code " : " last code sequences.

Graph: The selection dialogue



- The user may sort the rows by one or several columns by pressing on the column labels above the data cells. The box around the label will turn darker, and small arrow will show the sort direction. By pressing the CRLT-key, and clicking on a second column label, and second sort criteria is introduced. The size of the arrows shows the sort order. By continued clicks, the sorting order is first reversed, and then the original sorting re-established.

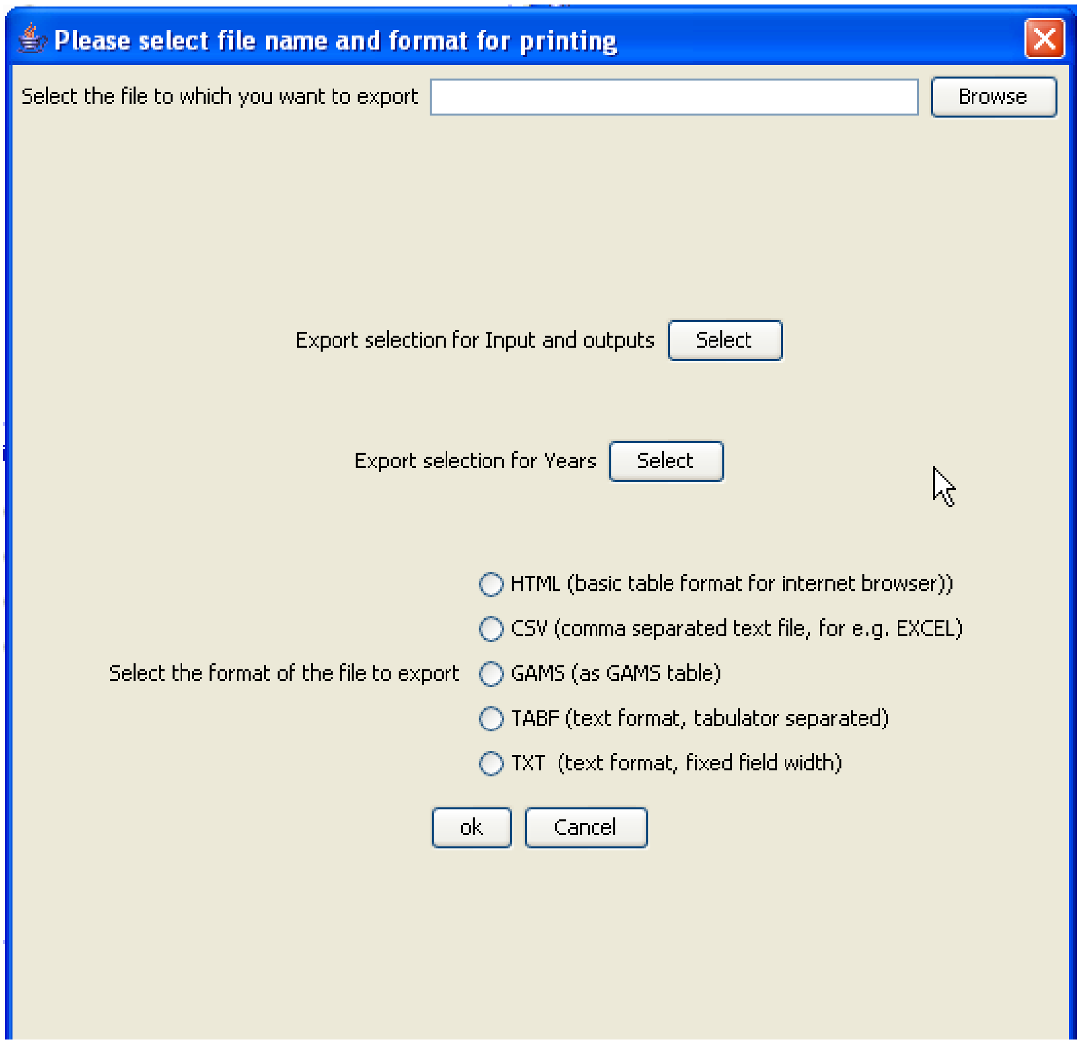
Graph: Sorting by data in a column



	1984	1985	1986	1987	1988	1989	1990
SI000000	8477.33	7788.77	7095.24	6399.62	5716.66	5084.81	4580.46
UK000000	7893.29	6596.76	7087.49	6182.38	6380.95	6849.00	7086.77
NL000000	7863.90	6694.80	8141.12	6996.71	7269.74	7545.09	7712.75
IR000000	7851.99	6475.53	5805.42	7379.31	8430.43	8092.67	9142.09
DK000000	7597.61	6009.86	6370.96	5963.27	6959.34	7463.51	7765.05
BL000000	6900.28	6306.18	6775.02	5565.24	6017.07	6769.55	6076.30
FR000000	6615.43	6120.29	5657.89	5731.91	6343.77	6532.46	6678.71
SK000000	6155.01	5810.10	5538.92	5475.46	5832.35	5587.64	5065.57
DE000000	5546.01	5737.46	5904.61	5997.89	5952.41	5683.90	6461.72
HU000000	5413.41	4942.81	4450.51	4508.64	5221.03	5206.63	4854.31
SE000000	5200.70	4594.75	5349.90	4609.87	4981.08	6060.78	6588.22

- The clipboard export will put the data, column and row labels as shown into clipboard, using tabulators as field delimiters. That format is accepted by most word processors, text editors, spreadsheet programs and many DBMS systems.
- The export dialogue offers additional options. Firstly, the user has to determine to what file the data will be exported where an existing file will be overwritten. Secondly, several formats are offered: rather simply formatted HTML-tables, GAMS tables, comma-separated files (CSV) and text format (fixed field width format). Selection dialogues for the dimensions not merged into columns and rows of the tabular view are offered as well.

Graph: The export dialogue



The work step "Generate base line"

For manifold reasons discussed in methodological papers, economic models as CAPRI are not suited for projections, but as tools for counterfactual analysis against an existing comparison point or an existing set of ex-ante time series. The point in time or these time series are called "base line" or "reference run". CAPRI "runners" which use the model for ex-ante policy simulation do not need to construct their own baseline, but are typically better off by sticking to the baseline provided on a yearly basis along with the latest version of the GAMS code, data base and software. Accordingly, the step and the included tasks are only for user type "administrator". According to current planning, the baseline will be updated in close cooperation with DG-AGRI twice a year in early summer and early winter, following the release of a new "medium term market outlook" by DG-AGRI.

The CAPRI baseline is a mix of trends, expert knowledge and automated checks for logical consistency, and is constructed by a sequences of tasks:

- *Generation of ex-post results*. Albeit not strictly necessary for the base line, the ex –post results often prove quite helpful when analysing the reference run. The ex-post results are model run for the base at base year policy and other exogenous parameters, inflated to the chosen simulation year.
- *Generation of policy shifts*. In order to capture in the later trend projection the effect of policy changes between the base and the simulation year, so-called policy shifts are calculated by applying the ex-ante policy to the state-of-world (technical progress, management practises, population numbers, per capita income, consumption patterns etc.) in the base year. The policy shifts are defined as the relative changes against the base year of implementing the ex-ante policy.
- *Generation of the trend projection*. The trend projection task is rather time consuming, and may run several days when the farm types are included. It consists of several sub-tasks. Firstly, independent trend lines for many different variables and all regions are estimated, and for each of these trends lines, statistics as R^2, variance of the error terms etc. are calculated. These results, together with the base period data and the policy shifts, are used to define so-called supports, i.e. the most probable values for the final projection. These sub-tasks are relatively fast. The final consistency sub-task is broken down in two iterations. In the first iteration, only the Member States consistency problems are solved. For the different projection years, the problem will look for minimal deviation from the supports – which may be interpreted as a priori information in a Bayesian interpretation – such that different necessary logical relations between the data are not violated – the data information in a Bayesian estimator. These relation define e.g. production as the product of yield and activity level or force close market balances. The details can be found in the methodological documentation. Once that step is done, the Member state are added up to the EU level, and new support are defined which take given expert projection into account, currently mainly a baseline provided by DG-AGRI. In the second round, the Member State problems are solved again, and then, problems for all NUTS II regions in each Member State, and, for all farm types inside of each NUTS II region.
- *Baseline calibration*. In the final task, the results from the trend projection serve as the major input to generate the baseline.

Running a simulation

At the core of CAPRI stands its simulation engine, which iteratively links different types of economic models: aggregate programming models at regional or farm type level, with an explicit representation of agricultural production technology, aggregated versions of these models at Member States model linked together to derive market clearing prices for young animals, and finally, a global spatial multi-commodity model for main agricultural products and selected secondary processed products.

Differences in results between simulation may be rooted in three different blocks:

1. Differences in the in-going base year data and baseline. CAPRI allows several base years and calibration points to co-exist, and user may choose the base and baseline year.
2. Difference in what economic models are linked together and in the regionalisation level as the user may switch the market modules on or off, may run the model at Member State NUTS II and farm type level or in comparative static or recursive dynamic mode.
3. And finally, the most common, differences in the exogenous assumptions including the policy definition.

Graph: The interface in simulation mode

Exploitation of gdx-files

GDX-files are generated by GAMS and typically serve either an exchange format between different GAMS applications, or for exploitation purposes as the GAMS-IDE comprises a view for GDX-files. Further tools for GDX-files are available from GAMS company and are described in different documents. In opposite to listings generated by GAMS programs, the GDX files store the data in full numerical precision in an internal format.

The new CAPRI version passes information from one task to the next with the help of GDX files, so generates CoCo a gdx files with the time series at national level, which is read by CAPREG. And the regional time series generated by CAPREG are inputted by the trend projection tool CAPTRD. These gdx files are accessed when the different tasks of “Data base exploitation” are chosen. The user has on top the possibility to load one or several tables from one or several freely chosen gdx files. When the task “exploit gdx files” is selected by pressing the related button, two buttons are shown in the task panel. The first one, labelled “load gdx files” will open a file selection menu when pressed. When the ok button of the dialogue is operated, the content of the gdx file is partially loaded, and a table is added to the right upper window of the application showing the parameters and sets comprised in the gdx files, along with their number of dimensions and records. When the close button next to the table is pressed, the table is deleted. Pressing the "load gdx file" again will add more tables.

One parameter from each table may be selected (pressing the "crtl" key when clicking with the mouse de-selects). If several parameters from one file needs to be loaded, the user may open the same file several time.

The content of the different parameters is merged together, and the parameters themselves span an additional data dimension. If the user does not provide input in the first column of the tables labelled "user input", the program will generate automatically names. The data loaded are shown in the table tool described above.

Software concept

History and general guidelines

The original concept of the CAPRI software concepts dates back to the year 1997 when the first EU financed project started. At that time, it was decided to use GAMS as the tool to build the core code of the economic model and to use template models. Templates models are structurally identical, i.e. they comprise the same equations and variables, but their instances differ in the parameters loaded into the equations. That concept is directly supported by GAMS.

Given the detailed break down of the economic models by NUTS II regions, many products and activities, a large data base became necessary to build all the necessary instances of the template model for the supply side, and it was decided to use a Data Base Management System to store statistical raw data, consistent ones and model results. The team at IFR Bonn had already existing tool and libraries at hand for DBMS and to build rather simple Graphical User Interfaces, build as a mix of FORTRAN and C-Code, in parts using a commercial portable GUI. These tools had already been successfully applied in the context of different economic modelling system, inter alia the RAUMIS system for Germany which was similar concept to the supply side of CAPRI, and the global trade model WATSIM which served as a blueprint for the first version of the CPARI market side. These decisions led to a a rather simple concept: the GUI let the user choose some simple settings (as for which regions and years to work), called upon the DBMS to retrieve the necessary input data, converted these data to GAMS tables and called a GAMS program as a sub-process which included the data and further use defined settings. The GAMS code in turn comprised statement to generate text file with the final results, and these were loaded back on the DBMS. In order to exploit the results of a run, the user had to rely on the FORTRAN/C based tools linked to the DBMS, mainly DAOUT, a multi-dimensional viewer with pivoting and exporting possibilities.

A first deviation from the general concept occurred at the end of the first project phase 1999 when part of the result data were stored apart in comma-deleted files to source the Java based mapping tool. Again, maps as such where nothing new, there had been FORTRAN/C based tools for maps in use since years in RAUMIS, whoever, the Java applet added much more user interaction as showing results for a specific regions when moving the mouse over the related polygon. Two years later, yet another outlet arrived with HTML-formatted input tables. Consequently, part of the result data were now already stored three times in different formats.

In 2003 it became obvious that the bell stared to ring for the FORTRAN/C solution. The underlying libraries were not thread safe, and using multi-processor machines resulted in rather frequent error messages and blocked processes. Trials to change the underlying C code yielded no serious improvements. Thus, it became clear that a new concept was needed. First prototypes of a Java based GUI and classes to replace the DBMS code were developed, at that time, where still several projects in house were using the old GUI concept, the aim was to replace the C-Code by Java while still keeping for an intermediate period the FORTRAN code, and then, gradually, move to a Java only solution for the GUI and DBMS. As almost all CAPRI users were using single-processor hardware and the C/FORTRAN solution continued to work under Windows XP, a rapid move to Java seemed not urgent, and the work at the Java solution stopped, as the growing use of CAPRI resulted in a high workload for the team in Bonn. The development process in Bonn concentrated on methodological improvements of the modelling system as the spatial trade model, ex-ante calibration, integration of the new Member States or the consistent link to the DG-AGRI projections, and not a new software concept.

The start of SEAMLESS along with the fact that more and more users worked on multi-processor systems and were thus facing an error-prone GUI pushed up the priority of a different software solution. Compared to the status in 2003 several facts had changed. Firstly, CAPRI was the only remaining system in Bonnn using the FORTRAN/C styled GUI, some older system were simply outdated, and some newer system had not yet developed an user interface. Accordingly, backwards compatibility with the old FORTRAN GUI libraries was not really longer an issue as the it less costless to develop a JAVA only version for CAPRI. Equally, it became obvious that the former well-funded arguments to use a home-made DBMS became obsolete: storage requirements and processing speed were not longer real bottle-necks given the powerful PCs available, and GAMS had generated its own intermediate file format called gdx which was faster and less storage demanding as text files and accessible via tools for export and viewing. On top, SEAMLESS carried the potential to build and maintain a GUI and DBMS targeted to large-scale modelling system. Not at least, any changes in the regional breakdown of CAPRI and its lists of products and activities or indicators required consistent structural changes both in the GAMS code, the DBMS and the FORTRAN code.

As a result of the facts mentioned above, the following decisions were taken:

- All CAPRI work steps and tasks will be steered from one application via a user interface, and not, as in the past, by several applications. The user interface shall be constructed as to support the work flow. The interface shall feature tools to track past work steps, and support access rights to applications differentiated by user groups.
- CAPRI will not longer feature a own DBMS, but rather store all intermediate and final results in gdx-files which are produced and read by GAMS code. The gdx-file can be accessed for exploitation purposes and in order to export data in another DBMS, as e.g. the SEAMLESS knowledge base, by JAVA classes provided by GAMS cooperation.
- The existing exploitation tools (maps, tables, graphs) will be re-written in JAVA and be sourced by gdx-files to prevent storing the very same data in different formats.
- The GAMS code will be changed such that the GAMS code generated by the user interface becomes minimal. All hierarchical relations between regions will be stored in GAMS sets, only.

Interaction with GAMS

The interaction with GAMS consists of three parts:

- Generating GAMS code based on user input
- Starting GAMS
- Controlling the GAMS run

There are two types of input files generated based on user input. The first one are so-called scenario files and define the exogenous drivers for a CAPRI run as population growth, macro-economic environment or policy definitions. Here, the final aim is to integrate the scenario editor from SEAMLESS into the CAPRI user interface. The scenario files are typically stored for longer period on disk, both to provide templates for other scenarios as well as for documentation purposes. The name of the file to load is passed to GAMS either as an argument or stored in a input file with a fixed name. The second types are rather small files with a fixed name which typically comprise the information for which years and regions to run the GAMS program along with a small number of methodological switches. These files are overwritten with each start of the related GAMS code.

GAMS is started as a sub-process in a own thread. The output from GAMS which is typically shown in command processor window is redirected into a pipe and its content read from there and shown in a window on the CAPRI user interface, so that the user can check GAMS execution at run time. The code allows to filter out specific statements generated by GAMS to be shown in the windows title bar to give an indication about program progress.

There are two final control mechanism. Firstly, the return code by GAMS which indicates if the GAMS program was correctly compiled and then executed. Typical execution time errors are math errors as division by zeros or read/write errors on external files. Secondly, the user can apply different type of exploitation tools to check the logical content of the results.

Exploitation tools

The structure of the GAMS generated gdx files

The exploitation tools load directly the gdx-files generated by the GAMS processes linked to the tasks described above. The gdx-files only store non-zero numerical values. The main content of a gdx file are two types of records. The first types provides a list of all labels used to identify the numerical data in the gdx file as GAMS does not support numerical indices, but requires character labels. The list does not distinguish for which data dimensions the labels are used, they are hence typically a mix of product, activity, region and further labels. The second type of records belong to GAMS parameters (scalars, vectors, or multi-dimensional tables). Each non-zero numerical item in each parameter has its own record. Each of these records provides the numerical data in double precision (depending on the parameter type there may be different data stored in one record, as for a variables its upper and lower bound, current level and marginal value etc.), and a vector of indices pointing in the list of codes described above.

Loading the data from gdx files

The data matrices generated by the different tasks as described above and stored in gdx-files are typically rather sparse, so that it seemed appropriate to load the data from the gdx-file into hash tables for exploitation purposes. That is done in a two step procedure. In the first step, all records from the gdx file are read and vectors of all found indices are stored. The length of each data dimension is only known when all data records are read, and is equal to the number of unique indices for each dimension. Once all records are read, the final length of these index vectors then defines a linear index room for the multi-dimensional table. In a second step, the records are read again, and the index vectors for each record now allow to define a linear index in the total table. A hash code is derived from that linear index to store the numerical values into a hash table. As the number of items to store in the hash table is known beforehand, a rather simple hash table implementation can be used. If necessary, step one can be run over several parameters which may be hosted in several gdx files, so that results can from different runs can be merged into one hash table.

As the gdx-files provide a lists of all labels used in any parameters stored in that gdx-file, the index vectors allow to build lists of labels linked for each index in a data dimension. There exists an additional storage type in the gdx-files to retrieve long-texts to the labels as defined in GAMS set definitions. However, one label may occur in different sets with different long texts, and the gdx-file do not store a possibly user defined relation between a data dimension of a parameter and a specific set, an option termed domain checking in GAMS. In order to link hence long-texts to the labels used for a specific data dimension, two options are possible. Firstly, the user may interactively at run time re-establish based on its knowledge the link between data dimensions and specific sets, and thus add long-texts to the labels used on that data dimension. Or the relation may be hard coded in the JAVA code.

Multi-dimensional viewer with pivoting and exporting possibilities

The multi-dimensional table is then loaded in a spreadsheet like viewer with pivot-possibilities. The user may switch between a tabular view of the data, or different type of graphs (line, bar, pie, spider). Scroll-down boxes allow the user to rotate through data dimension not shown in the view port columns and rows. Several data dimensions may be merged into one view port dimension. The user can use column and rows groups, and may apply selection to columns and rows as well as to columns and column groups. Rows carrying zero values only may be hidden. Rows may be sorted by size of the numerical values in one or several columns. The current table may be loaded into the clipboard. Alternatively, all or a selection of tables may be exported to a external file, in different format (HTML, CSV, tab-separated, GAMS, fixed width tables). There are further possibilities as changing fonts or the number of decimals. The viewer's concept is based on a tool already available as DAOUT in the old GUI, which should reduce the costs of CAPRI users to switch to the new solution.

A replacement of the XML/XSLT tables

Analysing the outcome of a large economic modelling system as CAPRI is not an easy task. A single model run will produce more then 1 Mio non-zero numerical items for a single year, and less experienced users feel lost where to start when analysing that data mountain. As an answer to that problem, the CAPRI team developed table views (as welfare analysis, market balances, prices or environmental indicators) which group together small selection of all data available. One may be tempted to see these as indicators – the major concept put forward by e.g. SEAMLESS to exploit results. Experience shows that there is not a clear distinction between raw model results and indicators. Rather, depending on the analysis at hand, different results and indicators are used in conjunction. Often, results for indicators must be explained by the underlying raw model results from which they are calculated. Here, the concepts of views helps the users as HTML-link draw logical connection between tables. So can the change in regional agricultural income tracked down in changed in income per activities and changes in activity levels. And the change in income per activity can be explained by changes in market revenues, premiums and production cost of each activity. And changes in market revenues are again resulting from changes in output coefficients and prices, and so forth. Interlinked tables showing these selected results hence allow the user a guided way into the huge data cube, and help to concentrate on major results and finding their causes.

In the very beginning, each of the tables consisted of hard coded HTML generated as the GAMS program executed, resulting in a huge number of files, loaded with formatting information. Further on, as several tables comprise the very same numerical results, these were stored several times in different tables. Any change in the structure or layout of the table required to run the model again, and all scenarios results had to be available at run time.

As a consequent next step, data and the structural and formatting information were separated. The data were stored in XML files and a XSLT program in conjunction with Javascript generated dynamically HTML based on a client side solution. However, the resulting program mix was successfully tested on Windows IE, only, and depending on how the data dimension where rotated in the view port, the XML parser had to parse a huge number of XML file to produce a HTML page with a few numbers, provoking long response times.

In order to reduce the number of different software instruments at hand, it was decided to embed the functionality of the XLM/XSLT in the Java based exploitation tool. That is done as follows. The table definitions (selection of items from each data dimension, related long texts, specific rotation) had always defined in XML and are now loaded by JAVA. The user may then scroll select at run time a table (as before in XSLT), and the multi-dimensional viewer will filter out the elements to show in columns and columns groups, rows and row groups. Some remaining issues (integrating the links, pivoting depending on the table definitions) are not yet incorporated in the current Java code and will be solved in due course. The latest version of the XML:/XSLT tables used Javascript to generate SVG-code to build different type of graphs. The majority of code is already successfully ported to Java and integrated into the table tool, however, the quality of the Adove SVG renderer exceeds the Java graphic engine.

The final version of the XML/XSLT table tool comprised code to generate SVG-codes maps. Albeit the graphical quality was quite high and exceeded that of the Java based mapping applet, the necessity to parse for each numerical item many XML-tables reduce the usefulness. One of the next steps is to port the underlying Javascript code to Java, and to provide a Java class to read the SVG-based polygon definitions.