# A New Graphical User Interface Generator for Economic Models and its Comparison to Existing Approaches

# Ein neues Werkzeug zur Erstellung Graphischer Oberflächen für ökonomische Modelle und sein Vergleich zu bestehenden Ansätzen

**Wolfgang Britz**
**University of Bonn, Germany**

## Abstract

*The paper discusses the role of Graphical User Interfaces in economic modelling and specifically the development of a new Graphical User Interface Generator for applications realized in GAMS (General Algebraic Modelling System), a widely used Algebraic Modelling Language for (bio-)economic simulation models. It motivates the development of GGIG (Gams Graphical Interface Generator) by reviewing existing approaches. In opposite to frameworks available for environmental and Agent Based Modelling, GUI generators for more classical economic models seem to be scarce. GGIG aims at a fast development process not requiring programming skills where simple user operable controls are defined in an XML file such that GAMS based projects can add easily a GUI. It comprises quite versatile exploitation tools for interactive reporting (tables, graphs and maps) including a machine learning package plus useful utilities, e.g. to generate a HTML based GAMS code documentation or batch execution. In opposite to other approaches, GGIG strictly separates the GUI and GAMS code, but does not offer IDE functionality. Applications to CAPRI, a rather complex model, and some smaller projects seem to show that researchers without formal programming training are able to develop and modify a GUI for their models, while model users can be quickly trained in GGIG based GUIs.*

## Key Words

*Graphical User Interface; Economic Modelling; Algebraic Modelling Language*

## Zusammenfassung

*Der Aufsatz diskutiert die Bedeutung Grafischer Benutzeroberflächen in der ökonomischen Modellierung, insbesondere die Entwicklung eines neues Werkzeugs zur Erstellung Grafischer Oberflächen für Anwendungen basierend auf GAMS (General Algebraic Modelling System), einer weitverbreiteten Algebraischen Modellierungssprache für (bio-)ökonomische Simulationsmodelle. Er motiviert die Entwicklung von GGIG (Gams Graphical Interface Generator) vor dem Hintergrund eines Überblicks alternativer Ansätze. Im Gegensatz zu bestehenden Programmiergerüsten für umwelt- oder agentenbasierte Modellierung scheinen nur wenige Werkzeuge zur Erstellung von Benutzeroberflächen klassischer ökonomischer Modelle verfügbar zu sein. GGIG zielt auf einen schnellen Entwicklungsprozess, der keine Kenntnisse in der Programmierung voraussetzt und auf der Definition einfacher Bedienelemente in einer XML-Datei basiert, wodurch sich ohne großen Aufwand eine Nutzeroberfläche zu einem GAMS-Projekt erstellen lässt. GGIG enthält darüber hinaus recht vielseitige Instrumente zur Erzeugung interaktiver Ergebnisauswertungen (Tabellen, Grafiken und Karten) einschließlich eines Paketes maschineller Lernverfahren sowie zusätzlich Dienstprogramme, z.B. zur Erstellung einer HTML basierten Dokumentation des GAMS-Codes oder zur Batchausführung. Im Gegensatz zu anderen Ansätzen trennt GGIG klar zwischen dem GAMS-Code und der Nutzeroberfläche, bietet dafür aber auch keine Funktionen einer Integrierten Entwicklungsumgebung an. Die Anwendung von GGIG in CAPRI, einem recht komplexen Modellsystem, und einigen kleineren Projekten legen es nahe, dass Forscher ohne formale Programmierkenntnisse mittels GGIG Benutzeroberflächen für ihre Modellen entwickeln und pflegen können, während Modellanwender sich schnell in die Nutzung der Oberflächen einarbeiten.*

## Schlüsselwörter

*Graphische Nutzeroberfläche; ökonomische Modellierung; Algebraische Modellierungssprache*

# 1 Introduction

The demand for quantitative policy assessments is currently growing, as governments increasingly face legal obligations for impact assessments (e.g. EU, 2009), which frequently involve the use of economic simulation models. One implication is that new staff, who was not previously involved in the development of a specific model, needs to be trained in model application or at least analysis. There are, however, not too many examples of policy-relevant economic models which succeeded in attracting users beyond the group of (original) developers and staff directly supervised by them, such as graduate students. GTAP (HERTEL, 1997) is probably the best-known case. Another example for the transition from development and use by developers to continuous policy-relevant application by non-developers provides the so-called iMAP (integrated Modelling Platform for Agro-Economic and Policy Analysis, M'BAREK et al., 2012) platform. The platform hosts both partial equilibrium models such as ESIM, BANSE et al., 2005; CAPRI, BRITZ and WITZKE, 2012, and AGMEMOD, SALAMON et al., 2008, and several Computable General Equilibrium models (GTAP; MAGNET, VAN MEIJL and WOLTJER, 2012; GLOBE, MCDONALD et al., 2012; RegCge, BRITZ, 2012). iMAP aims at providing a scientific basis for policy decision-making linked to the economic assessment of the Common Agricultural Policy and related topics such as trade, energy, environment, and climate change.

The reason why certain models, such as those mentioned above, are used continuously, also beyond their developers, or are proposed by clients for applications while others might never provide policy-relevant results clearly depends on many aspects. These include a sound methodology; appropriate extent and resolution in space, time and with regard to products and processes; the provision of policy-relevant indicators or detail in depicting policy instruments (cf. PODHORA et al., 2013). Software aspects (BRITZ, 1999) also play a critical role, specifically with regard to software usability (cf. ABRAN, 2003). ISO9241-11, 1998 defines software usability as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". Software usability clearly affects learning costs and, therefore, the ease with which new model users can be trained.

The paper focuses on these technical aspects and specifically on the role of Graphical User Interfaces (GUIs). Beyond a more general discussion on the use of GUIs for economic models, partly drawing from the examples mentioned above, the paper also presents a new freeware tool to build GUIs for economic models, currently available for models written in GAMS (General Algebraic Modelling System, BROOKE et al., 1988) or the statistical language R (IHAKA and GENTLEMAN, 1996). A structured discussion on experiences with that tool named GGIG (Gams Graphical Interface Generator, BRITZ, 2014a), and its predecessor, the CAPRI GUI, provide further empirical evidence for a more general discussion. The paper is organized as follows: the next section discusses selected software aspects. Section 3 analyses the role of GUIs and discusses options available for GAMS-based models for model steering and result analysis, before presenting GGIG in section 4. Section 5 provides a short comparison to alternative solutions and discusses experiences with GGIG and the CAPRI GUI. Finally, a summary is provided in section 6.

# 2 Selected Software Aspects in Agricultural Economic Modelling

Partial and general equilibrium models at a larger, often global scale, but also bio-economic farm and hydro-economic models typically use ALMs (Algebraic Modelling Systems, KALLRATH, 2012). KALLRATH (2012) describes an ALM as "Roughly speaking, a modelling language serves the purpose of passing data and a mathematical model description to a solver in the same way that people, especially mathematicians, describe those problems to each other". AMLs are especially useful if data transformation and equations are structurally identical, e.g. across regions and products. Table 1 presents technical details on some well-known policy relevant models in agricultural economics. The majority of the models use either the AMLs GAMS or GEMPACK (HARRISON and PEARSON, 1996), with GEMPACK being far more specialized and targeted to CGEs. Econometrically estimated economic models such as AGLINK-COSIMO or FAPRI often simulate within the same econometric package used for estimation.

The core of an economic simulation model encoded in an ALM consists of numerical problem(s) that require a simultaneous solution for all equations. This contrasts with environmental models that often

**Table 1. Software aspects of selected models relevant for agricultural policy/market analysis**

| Model | Short characterization | Simulation language | GUI |
|---|---|---|---|
| GTAP | Global, trade oriented CGE | GEMPACK (GAMS version available) | runGTAP (PEARSON et al., 2003) |
| MAGNET | GTAP variant with a focus on Europe and the CAP | GEMPACK | Proprietary (WOLTJER, 2013) |
| GLOBE | Global CGE | GAMS | - |
| ESIM | Multi-Commodity model, strictly template based | GAMS | - |
| AG-MEMOD | Multi-Commodity model, single equation based | GAMS | GSE based (SALAMON et al., 2008) |
| AGLINK-COSIMO | Multi-Commodity model, single equation based (COSIMO template based) | TROLL | Visual Basic/EXCEL based |
| FAPRI | Multi-Commodity model, single equation based | EXCEL, SAS (cf. MOSS et al., 2011) | Result analysis in EXCEL |
| CAPRI | Regional/Farm type programming models linked with Multi-Commodity model and regional CGEs, all strictly template based | GAMS | GGIG |

Note: BRITZ et al. (2013) provide a detailed comparison of the result exploitation part of GTAP, AGLINK-COSIMO and CAPRI. Where no reference is given, information is based on personal communication with model authors.

integrate smaller components and are solved recursively in space and time, which asks for a modular design. That might partly explain the evolvement of frameworks for environmental models (LANIAK et al., 2013a), often targeting specific application domains (LANIAK et al., 2013b) and with a focus on interoperability of individual modelling components. These frameworks typically support tool development including GUIs as well as result visualization[1]. Similar packages (cf. RAILSBACK et al., 2006) exist for Agent Based Modelling (ABM). Compared to economic models, the resolution of environmental models in space and/or time is often higher, but the number of items simulated tends to be smaller. With dynamic and/or spatial aspects often in focus, time series graphs and/or maps are widely used for visualization.

Contrary to that, the simultaneous solution approaches in economic models tend to combine all variables and their relations present in the overall problem in one module. CGEs and Multi-Commodity models solve for many items such as different types of prices, output generation, primary factor and intermediate input use, trade, demand categories, differentiated by product and space, and eventually time. All CGEs and also the partial equilibrium models (ESIM and CAPRI) mentioned above are therefore rather strictly build in a template structure. This means that model equations are structurally identical across space and products, and where applicable across periods, while differences are expressed in parameters. AMLs allow efficient coding of such template based models. For economic models, tabular presentations or bar charts with relative changes are widely used to depict results, in order to assess the linkages across markets and items. However, also economic models with a higher spatial resolution often use maps for result visualization, whereas (recursive) dynamic models tend to rely on time series graphs. BRITZ et al. (2013) provide a comparison of GUIs with a focus on result visualization such that these aspects are not discussed in detail in the following.

Besides differences rooting in the solution strategy, software solutions supporting modelling in different disciplines reflect different traditions to document and describe models. Economists rely mostly on equations, and, therefore, favour AMLs, whereas environmental modellers more often additionally use graphical presentations, such as flow charts. They might, therefore, favour object-oriented approaches or even frameworks that allow building models with the help of GUIs (cf. RICHMOND and PETERSON, 1997). Thus, both software use and result analysis

[1] The thematic issue on "The Future of Integrated Modeling Science and Technology" of Environmental Modelling and Software, Vol. 39 provides an excellent overview on the current state-of-the-art and visions in that field. It is interesting to note that very few of the 26 articles in the thematic issue touch also upon economic optimization models (e.g. KNAPEN et al., 2013, on the use of OpenMI, BULATEWICZ et al., 2013, on integrating scripting languages such as MATLAB into OpenMI which also more widely used by economists).

are linked closely to structural properties of the underlying models as well as disciplinary traditions. However, path dependencies can also play an important role.

In agricultural economics, the AML language GAMS is widely applied (BRITZ and KALLRATH, 2012), also for tools with a focus on environmental interactions such as bio-economic single farm models (cf. JANSSEN et al., 2010) or global land use models (cf. HAVLIK et al., 2013). Contrary to many packages used in environmental and ABM modelling, GAMS is a commercial, non-open source product. It provides transparent interfaces to solvers and supports a compact, set-driven presentation of data transformation and models, along with a rather powerful scripting language. GAMS is shipped with a basic Integrated Development Environment (IDE, a software tool which supports code development by e.g. specialized editors) and simple tools to inspect the content of proprietary data bases. GAMS does however not comprise a GUI generator. Commercial GAMS applications in different domains[2] such as electricity grid optimization (ANG, 2004), portfolio optimization for electric utilities (REBENNACK et al., 2010), gas transmission optimization (DE WOLF and SMEERS, 2000) or chemical engineering (MORARIA and GROSSMANN, 1991) are often integrated in a software environment already featuring a GUI. Therefore, it might not pay off for GAMS to develop a GUI generator (see also BRITZ and KALLRATH, 2012); instead, GAMS offers Application Programming Interfaces (APIs)[3] which allow integrating GAMS into other software frameworks.

Therefore, there exists no default GUI solution for GAMS-based tools. That leads to proprietary solutions such as BAZZANI (2005) for a water management tool or the GUI (BRITZ, 2011) for CAPRI (Common Agricultural Policy Regionalized Impact Model, BRITZ and WITZKE, 2012). BRITZ et al. (2013) review, with a focus on result analysis, GUI solutions of three large-scale economic simulation models - partly realized in GAMS - and conclude that the development of common GUI tools might be advantageous, to avoid costly duplicate coding efforts and combine efficient and innovative solutions of existing GUIs. However, not much is available in the market in that respect. To the best knowledge of the author, so far only DOL (2006) has developed with GSE (Gams Simulation Environment, cf. DOL and BOUMA 2006) a generic tool with GUI functionalities for GAMS models. Other approaches, such as SEAMLESS-IF (VAN ITTERSUM et al., 2008; KNAPEN et al., 2013) or SIAT (VERWEIJ et al., 2010) – which wrap a layer around a GAMS application to integrate them into an Open-MI based architecture – target larger modelling tool for integrated assessments which incorporate different components.

But is it possible to develop a generic GUI generator for (bio-) economic models using GAMS which is on the one hand easy enough to handle for GAMS coders while on the other hand flexible and powerful enough to make its use attractive? What are the alternatives? Does it pay off to add a GUI to GAMS-based tools? Against this background, this paper documents and discusses a new interface generator for GAMS and R-based applications termed GGIG, coupled to exploitation tools and additional utilities such as for HTML based code documentation. In the next section, its development is motivated and main functionalities discussed, drawing a comparison to alternative solutions.

# 3 GUIs for GAMS-based Economic Models

## 3.1 Why a GUI?

Since GAMS is not shipped with a GUI generator, the default solution to steer GAMS-based applications consists of using a text editor to change specific sections of the code. To do this, users need familiarity with GAMS and detailed knowledge on the code structure of the economic model. Here, GUIs can decrease learning costs for users which need not to familiarize themselves with the details of the underlying code (cf. ABRAN et al., 2003). Such detail refers e.g. to the mnemonics used in the code – the names of symbols and labels –, and the file structure. Indeed, traditional training for specific economic models often comes close to learning a new language. In that case the trainee needs to memorize a new vocabulary of parameter, variable and equation names as well as labels used for products, regions and items; often in combination with a new grammar, i.e. a software language such as GAMS and how it is specifically applied in the model. A well-designed GUI introduces a layer accessible to users which shields details of the technical implementation from the user such that the

[2] Publicly documentation of commercial applications is scarce, possibly as the detailed knowledge constitutes a competitive advantage.

[3] see http://www.gams.com/dd/docs/api/

conceptual knowledge about the economic model is sufficient for its successful application. Additionally, it is typically faster to operate controls on a GUI compared to changing GAMS code with a text editor. Learning costs for a GUI are typically low if it is based on the look and feel of a normal windowed application.

Decreased learning costs might be important in a university or other research environment with a higher staff turnover, to allow e.g. graduate student to apply models or to contribute to their further development. That especially matters if model use expands beyond the team of (original) developers to economists involved in policy-relevant applications. As such, introducing a GUI might be part of the transition process of moving tools from development to application.

Steering a complex model by manual edits directly in the code also increase the chance of errors, especially if steering options are distributed across files: the operator might have forgotten to reverse changes from earlier runs or to introduce all the changes necessary for the intended application. Contrary to that, a GUI shows all options available (hopefully in a appropriately ordered and compact way) on the interface itself. A GUI can also prevent steering errors by restricting input choice, e.g. by attaching numerical ranges to input editors. GUI generators might also offer additional specific utilities targeted to the use of GAMS or another language. Available GUIs for GAMS tools also comprise reporting tools which can speed up debugging, analyzing and publication of quantitative results. That might hence open the chance for class-room use of more complex models, typically hard to realize if results can only be accessed from inside the modelling software.

Adding a GUI to a tool can also support code development. Constructing a GUI forces the coder to clearly define which settings the user can change for a certain type of application, along with their allowed ranges. Equally, a GUI will typically pass these settings in one block to a language such as GAMS, such that ideally only one file changes between runs which supports a clear separation of input and software code. It also eases the use of a software versioning system, as run specific edits in a whole set of files are avoided.

GUI development can also help to reflect more clearly how to modularize the production chain of an economic model (data input and transformation, preparation of an ex-ante baseline, parameter estimation or calibration, scenario definition, simulation runs, post-model processing, exploitation), and to define clear input-output relations between these different steps. Producing output reports of each step which can be inspected by the GUI supports quality management through introduction of logical breakpoints where (intermediate) results are inspected. Efforts to support a clear code structure are specifically important in GAMS projects, as the language only allows for global symbols and basically does not support sub-routines or functions.

Most of the reflections above are not specific to GAMS, but hold for any economic simulation model, either implemented in another AML, an econometric package or another computer language. Indeed, GGIG, the package discussed below, now also supports project results in the econometric package R and was also linked to a Java-based ABM, whereas components of GSE, i.E. a similar system, can be used with GAMS and GEMPACK.

## 3.2 Options to Steer GAMS-based Tools and Exploit their Results

The following section briefly compares four basic options found in practise to steer GAMS-based tools: (1) no GUI, (2) simple EXCEL/Visual Basic based solutions, (3) GUI generators specifically designed to interact with GAMS and (4) proprietary solutions for a specific tool.

The perhaps most widely used approach is no GUI at all, i.e. to steer the GAMS application by adding or changing settings directly in the code. The disadvantages were already mentioned above. It has clearly the lowest development costs and does not require knowledge beyond the one needed to develop the GAMS code itself. Of the models mentioned above, only ESIM and GLOBE seem to rely on that solution. It might be the appropriate one if only developers already familiar with the code use the model and/or if the code base is relatively small and there are not many settings to change, such that the learning costs to oversee the full code are small.

A proprietary GUI, i.e. one specifically developed for one tool, gives the highest flexibility with regard to layout and functionality, but is also the most costly alternative. Most research groups developing GAMS code are not familiar with the progamming languages, libraries or GUI builders needed to develop a GUI. This means that external expertise must be hired, which can lead to substantial

transaction costs. More over, economic models are typically permanently updated, for example in response to (proposed) policy changes. That might provoke changes in GUIs to reflect modifications in the GAMS code. As updates take time, GUIs coded by third parties typically lag behind the model's code development. Indeed, cost and time to synchronize changes in the model's code with a GUI is one possible reason why GUIs attached to economic models were sometimes given up again (cf. BRITZ et al., 2013). That clearly motivates a solution where GAMS coders can generate and change the GUI themselves. Of the models mentioned above, GTAP and MAGNET use a proprietary GUI. CAPRI has also over years been based on a proprietary solution, and AGMEMOD as well features its own user interface (VAN LEEUWEN et al., 2012), which is however closely linked to GSE. All these systems are rather large, and receive(d) considerable funds for development and maintenance, which might explain why proprietary solutions could be developed and kept alive.

Packages such as GSE and GGIG are specifically designed to build GUIs for GAMS based tools. They are thus less flexible compared to a proprietary solution, but require far less time and limited knowledge to build a GUI. They also aim at allowing the GAMS coders themselves to synchronize code and GUI development. The use of that type of GUI does not necessarily require GAMS knowledge. Besides the examples discussed below based on GGIG, the CCAT tool (Cross Compliance Assessment Tool) project (BOUMA et al., 2010) realized in GSE provides an example about GUIs built with such as GUI builder.

Alternatively, tool developers have developed solutions building on a GAMS<->EXCEL interface delivered with GAMS. Users edit numerical values in predefined EXCEL cells, from where their input is read by GAMS. Combined with a Visual Basic application which starts GAMS, that allows building of a rudimentary GUI for GAMS model. Visual Basic would also allow to introduce user operable controls and check their input. A similar solution is reported for AGLINK-COSIMO (OECD, 2007), which is realized in TROLL, an econometric language which also does not feature an own GUI-builder. These solutions tend to deliver a Look and Feel distinctely different from a normal Windows program, while requiring expertise in Visual Basic programming.

# 4 GGIG: A GAMS Graphical Interface Generator

## 4.1 Background

Since 1999, a Java based proprietary GUI is available for CAPRI, a large-scale, global agricultural economic model with a focus on Europe which comprises also environmental modules, including a spatial down-scaling component (LEIP et al., 2008) covering 150,000 1x1 km clusters linked to bio-physical modelling (BRITZ and LEIP, 2009). GAMS is used for basically all numerical operations in CAPRI: data fusion, model set-up and solution as well as post-model processing. The GUI attached to CAPRI emerged slowly over years. Its exploitation part is based on interactive reports (tables, graphics and maps), produced from multi-dimensional parameters read from GDX files, a proprietary binary format from GAMS for which APIs are available. These reports are defined in an XML file which rendered the exploitation part generic enough to use it already in the past for e.g. an economic agricultural model for Benin (KUHN et al., 2010) or a Morrocan river basin model (HEIDECKE and HECKELEI, 2010). Drawing on the positive example of the XML-based report generator, the aim of developing GGIG was twofold: first, to provide a new, generic concept for GUI generation for GAMS based tools which can be applied not only to CAPRI, but also to other GAMS-based economic models. That included the goal to overcome certain disadvantages of the proprietary solution available for CAPRI. Second, to integrate functionalities developed originally only for CAPRI into the GUI generator in order to port them also to other projects using GAMS.

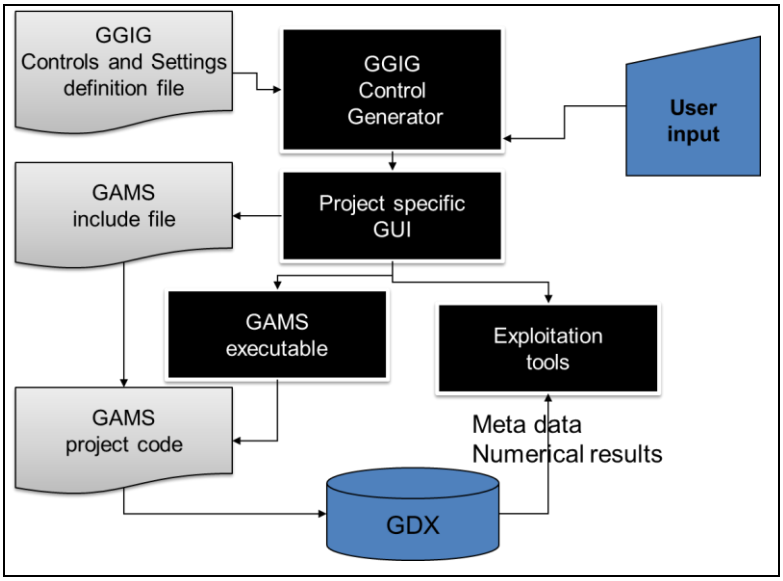
The development of GGIG and the preceding CAPRI GUI were thus not based on a formal user requirement analysis. Instead, core CAPRI GAMS code developers developed its GUI in parallel to the model code and improved it over time based on feature requests and feedback by CAPRI users. The annual CAPRI training sessions where both core CAPRI users and newcomers come inter alia together to analyse scenarios based on the GUI provided a forum to both get feedback on the GUI and to observe how people (learn to) use the GUI. The somewhat informal development process also reflects the fact that there are no commerical interests involved as the GGIG binaries and CAPRI[4] are distributed for free.

[4] Further information on GGIG along with a fully operational downloadable didactic example can be found at:

## 4.2 General Concept and Current Applications

Two major object classes underlie the general concept of the steering part of GGIG[5]: (1) **tasks** which are linked to GAMS applications, and (2) **user operable controls** (checkboxes, sliders, spinner, single and multi-lists, editable tables, text fields, file selectors), derived from standard Java Swing components (cf. ROBINSON and VOROBIEV, 2003), typically shared by several tasks. GGIG transforms the state of these controls into a standardized presentation in GAMS language constructs, sent to a GAMS application based on a file to be included (see figure 1)[6]. The GUI developer defines objects of these two classes in a XML file – the GGIG controls and settings definition file – from which GGIG builds the GUI. The user can execute a task as a GAMS application from the GUI which also shows run time messages from GAMS. The reporting part then allows merging of results from different runs and exploiting their results. As discussed below, GGIG offers additional utilities for working with GAMS-based models.

**Figure 1. Overview on information flow in GGIG**




Source: own elaboration

The overall layout of the GUI is standardised and unchangeable, see Figure 2. It comprises the following main elements: (1) a menu bar to change project-wide settings (such as directories, the GAMS version to use, SVN related information, user name and type) and to access utilities (discussed below); (2) a work step and task selection panel on the left hand side; (3) a right hand side panel which either shows: (a) controls, a button panel to start GAMS and a windows capturing GAMS output, (b) a panel to select result sets (see the left hand side of Figure 5 below) and to start their exploitation or (c) the exploitation tools (see the example on the right hand side of Figure 5).

The project specific XML file defines the work steps, tasks and controls available to the user. Equally, the reports shown in the exploitation tools are task specific. Whereas the grouping of the controls is rather free, the fixed standard layout underlying GGIG forces the GUI developer to structure its GAMS project into work steps and tasks. That is a deliberate restriction guiding also the code development of a tool such that it reflects a logical sequence of steps, for example, data base compilation, baseline development, model calibration and counter-factional runs.

As opposed to Java GUI generators such as WindowBuilder[7], GGIG offers quite limited layout possibilities and no WYSIWYG (What You See Is What You Get). That reflects the aim to let model users develop a simple GUI for their own purposes. The declaration of a control, therefore, comprises solely a few attributes, which reduces learning costs (see Figure 3 which shows the most commonly used ones for an example).

## 4.3 GAMS Application Steering

As discussed above, the user steers GAMS applications by operating the controls provided by GGIG. Specific options for controls in GGIG should help to reduce steering errors. First, the input for any control requiring numerical input can be restricted to a predefined input range; file selection can be limited based on a REGEX mask. Second, there is an n to one relation between GAMS applications (= mains) and a task

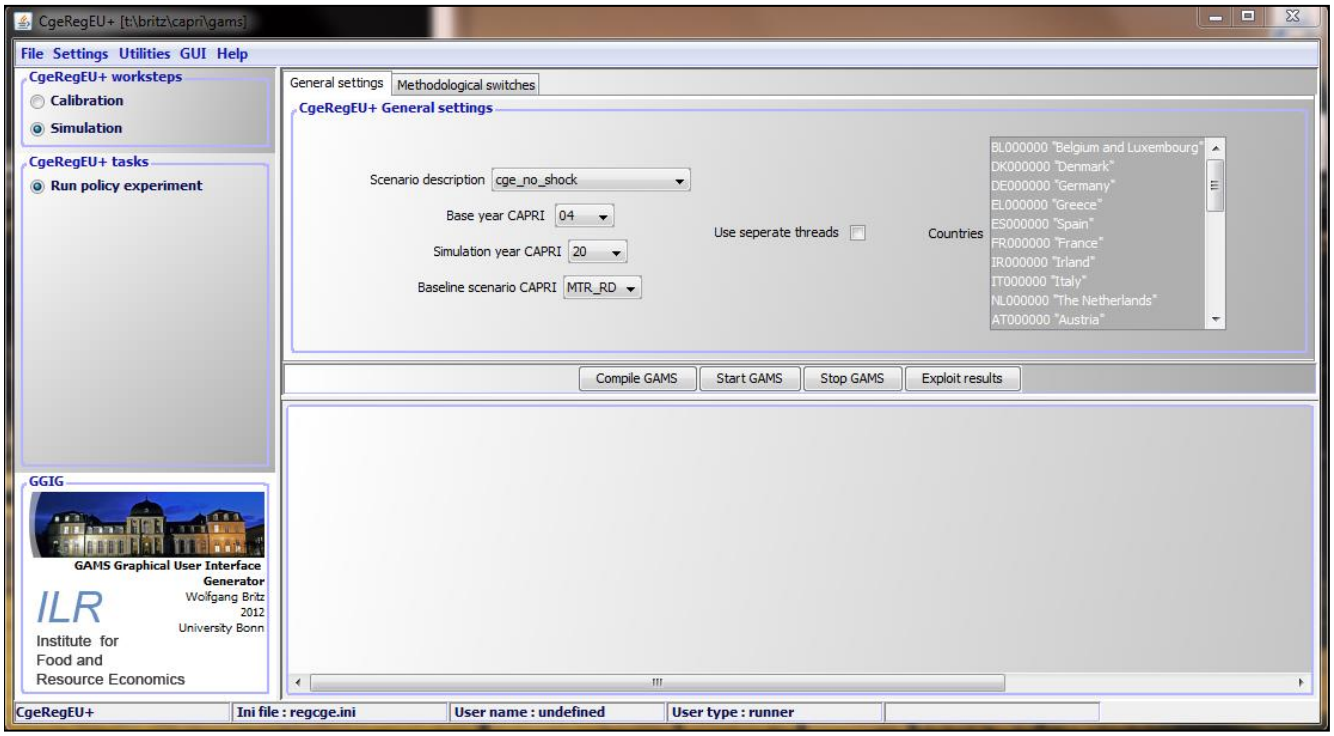
http://www.ilr.uni-bonn.de/agpo/staff/britz/ggig_e.htm. For information on CAPRI see: http://www.capri-model.org.

[5] The following section draws to a large extent on the GGIG manual, written by the author, from which also figures are copied.

[6] The GUI can also be used to steer tasks in the econometric package R, in which case R code is generated from the control setting, such that it can provide a GUI for tools project which use both R and GAMS.

[7] WindowBuilder is a GUI builder for JAVA which works as an Eclipse plugin (http://www.eclipse.org/windowbuilder/).

**Figure 2. The main window of a GGIG based GUI**




Source: screenshot from GGIG generated GUI

in GGIG. If an application is used in different "modes", e.g. for model calibration versus simulation, each mode can receive an own task and thus set of controls. That is also the only way to introduce relations between controls in GGIG. Third, default value(s) for each control can be registered and, fourth, depending on the user level (exploiter, runner, administrator, developer, debugger) tasks and controls can be hidden or disabled. Indeed, under the exploiter level, a user can solely exploit results and does even not see any control. The settings of all controls are sent in one block to GAMS. That renders it easy to write GAMS code for more complex crosschecks across control settings and to report the error back to the user from GAMS.

With the exemptions of tables and n from m selection (which require to pass a vector or matrix of settings as an appropriate GAMS symbol), all other settings are passed as a "$ETGLOBAL key value" pair to GAMS. That gives the coder high flexibility in handling the settings in GAMS. Additionally, all settings are passed as strings via a SET declaration to GAMS (see Figure 4) which provides thus a complete meta-information on the run. The application can store that information with numerical results in a GDX container for later inspection. The GUI also remembers the state of the user operated controls and further input between sessions.

**Figure 3. Example for a XML definition of a control in GGIG**

```
<control>
        <Type>checkBox</Type>
        <Title>Downscale farm type results</Title>
        <Value>false</Value>
        <gamsName>useFarmTypes</gamsName>
        <tasks>
             HSMU baseline,
             Downscale scenario results
        </tasks>
</control>
```

**Figure 4. Example of meta information generated by GGIG and passed to GAMS**

```
SET s_META /
'Workstep' 'Run scenarios'
'Task' 'Simulation'
'Date and time' '2013-05-08 14:59:48'
'Scenario description' 'NoHumans'
'Number of years' '5.0'
'Water rights, updated according to lake level' 'on'
'Solution printing' 'Suppress'
 /;
```

## 4.4 Exploitation

The exploitation tools offer different types of views: tables, graphs and maps. The user can add relative or absolute differences to element(s) in one or different dimensions and export views to clip

board and different file formats. Tables support e.g. pivoting, filtering, sorting, basis statistics and showing statistical outliers and allow adding long labels, units, pop-up explanatory texts and links to a section in a PDF file. Table items can carry a hyperlink, e.g. for navigating through a sequence of tables with increasing detail. Additionally, users can for example choose fonts and number formatting. The graphs (e.g. line, bar, box-and-whisker and spider charts, heat maps and histograms) build on the JFreeChart[8] library and are user configurable, for example with regard to fonts, transparency or colouring. Maps allow for different classification and colouring options, and support some more unusual formats such as flow maps (see Figure 5) or bar charts embedded in a map. A utility allows importing geometries in Shapefile format for use with the mapping viewer. For a detailed description of the exploitation tools see the GGIG (BRITZ, 2014b) and CAPRI (BRITZ, 2014c) user manuals. Tasks in GGIG can define filters when selecting result sets from a disk; Figure 5 presents an example taken from the CAPRI GUI, which also shows a flow map.

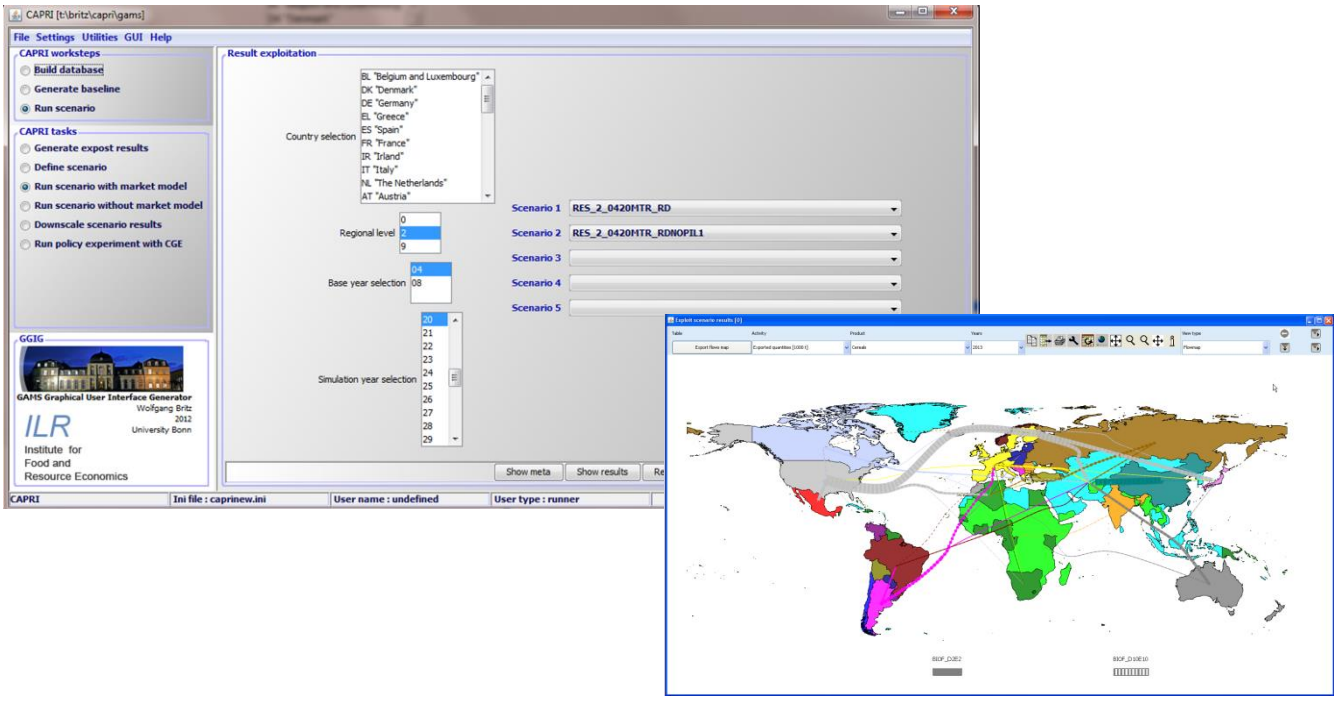
Additionally, the exploitation part transparently integrates the WEKA machine library (WITTEN et al., 2011). It provides a powerful set of filtering, clustering and classification algorithm as well as related visualization tools from machine learning, and allows for a data driven approach to explore relations between results loaded in the viewer.

## 4.5 Further Functionalities

GGIG supports further functionalities originally developed for the CAPRI GUI and now generalized to work with other GAMS projects: (1) batch mode, (2) automated code documentation in HTML, (3) SVN updates, (4) a viewer for GDX files and (5) an equation and variable viewer for GAMS models.

The first two might deserve some further explanation. The batch mode allows running a sequence of tasks without using the GUI, such as different simulation runs, while it documents in a HTML file which tasks were started, their settings and return codes. The generated include files and the GAMS listings of each task are saved for later inspection. The underlying batch steering file can be constructed by copy and paste from generated include files. The code documentation generator generates a HTML site with interlinked pages for each GAMS file, GAMS symbol and tasks, similar to javadoc[9]. That allows, for example, finding out for a certain parameter or variable in which tasks and files it is defined, used or changed. The HTML page also collects SVN information on each file.

**Figure 5. Example of a result selection panel and map view generated by GGIG**




[8] http://www.jfree.org/jfreechart

[9] Cf. http://agile.csc.ncsu.edu/SEMaterials/tutorials/javadoc/

# 5 Discussion

## 5.1 Short Comparison of GGIG to Modelling Frameworks and other GUI Generators

Compared to frameworks used in environmental modelling or the libraries used for ABMs, GGIG serves a far more limited task: it allows solely adding some basic GUI functionalities to a GAMS project. It does not support interactive graphical development of model code or a graphical GUI generator. Furthermore, the interaction between GGIG and the GAMS code of the model is rather limited: GGIG solely generates a typically rather small include file, spawns the GAMS application as a separate process and shows it run-time output on screen. Finally, it can read results from several GAMS runs from disk to exploit them. Therefore, GGIG does not pass objects in memory back and forth to the model. Accordingly, GGIG typically requires very limited changes to the GAMS code, such that the GAMS application can still be used alternatively within the GAMS IDE or started by another application. CAPRI, to give an example, was steered in the context of another project by a client-server based GUI (RIZZOLI et al., 2009). GGIG is portable between those platforms supported by both GAMS and Java.

Perhaps the product that is most similar to GGIG is the GSE tool by DOL (2006). Both target GAMS projects, but their basic concepts seem rather different. GSE combines functionalities of an IDE (e.g. editing, visual presentation of links between files, version control), a GUI and a reporting tool. Overall, GSE seems more powerful, but learning costs and the effort to implement GSE into a project are probably also higher. Whereas GGIG is steered by XML files to define controls, tasks and the reporting views, the GUI in GSE seems to be mostly set up by introducing steering tags as comments in the GAMS code. GGIG might be somewhat more versatile with regard to exploitation possibilities.

Similar to runGTAP, a GUI developed for the GTAP modelling system, GSE keeps a strong link between the data structure present in the GAMS code and the visualization. That clearly fits a more IDE-focused profile. The exploitation tools of GGIG, in contrast, are set up to work more similar to a data mining tool, i.e. the reports are not (necessarily) structured according to data structures present in the underlying GAMS model. That somewhat increases learning costs for that part of GGIG, and, in order to fully exploit the potential of the exploitation tools, requires an appropriate structuring of the results in GAMS.

## 5.2 Experiences with GGIG

Examples of projects which use GGIG beyond CAPRI, in chronological order, provide a small partial equilibrium model for the global poultry market (WIECK et al., 2012); FARMDYN, a detailed single farm bio-economic model (LENGERS and BRITZ, 2012); a hydro-economic river basin model (KUHN, 2012) and a prototype GAMS version of the AGLINK-COSIMO model. Furthermore, the FADNTOOL project (http://www.fadntool.eu) has recently opted to use GGIG for a tool that combines different economic models, partly realized in GAMS and partly in R, which all use Farm Accounting Network Data. Table 2 reports some key properties of these current projects known to the author where GGIG is applied. It shows that the use cases and thus the aims linked to the application of GGIG somewhat differ. Equally, it highlights larger differences in the complexity of the generated GUIs when assessed by the number of tasks, controls and views defined in the XML files steering. A view as reported below refers to one predefined table, map or graph to exploit results.

Based on these examples, experiences with GGIG so far will be reported and analyzed for three use cases of increasing complexity (see Table 3): (1) analyzing results, (2) model runs, and (3) setting up a GUI interface and extending it. The table below classifies these use cases with regard to the knowledge needed and tries to generalize to what extent a GUI might reduce learning costs, referring to arguments from section 2. Details for the use cases will be discussed next.

The *first use case* refers to *analysis of existing results*. The user does not need to run the economic model, which also means in the case of GGIG that no GAMS or R installation is necessary, only a working Java Run Time Engine (JRE) is needed. GGIG allows (pre)selecting a user role "exploiter" where all model steering controls are hidden. Only such tasks are visible for which existing result sets can be located. Accordingly, the user only sees drop-down boxes to select result sets, e.g. from different scenario runs, and, if defined in the XML file, selection box e.g. for years or countries. If locations on disk are relative in presets, one can copy a work installation of a GGIG GUI along with result files e.g. from USB stick to a computer, and the user can immediately start the interface, as long as the JRE is correctly installed. That renders

**Table 2. Properties of projects applying GGIG**

| Project | # of tasks (T), controls (C) and views (V) | Users | Main aim of using GGIG |
|---|---|---|---|
| CAPRI (large scale partial equilibrium model) | T: 25, C: 145 V: 182 | developers, runners, analysts | Replace proprietary GUI to allow GAMS coder to change GUI |
| Global poultry equilibrium model | T: 1, C: 20 V: 16 | developers | Test of GGIG, ease result analysis |
| FARMDYN, detailed bio-economic farm model | T:4, C: 173 V: 28 | developers | Support modular code development, ease model application and result analysis |
| AGLINK-COSIMO in GAMS | T: 11, C: 28 V: 28 | developers, project reviewer | Support modular code development, ease testing, show that GAMS based model implementation can be easily linked to GUI |
| LANA-HERBAMO (river basin model) | T: 2, C: 19 | developers | Ease model application and result analysis |
| GTAP in GAMS with GGIG | T: 6, C: 28 | Students as developers, runners and analysts | Ease coding, model application and result analysis |

Source: own research

**Table 3. GUI use cases and related knowledge w/o GUI**

| Use case | Knowledge needed independent of GUI use | Contribution of GUI to reduced learning costs | Additional knowledge needed without GUI |
|---|---|---|---|
| Model results analysis | Market and policy intelligence, economic theory, methodological overview on model | Potentially very high | At least mnemonics and file location/structure of results |
| Model runs | Additionally: details on methodology | Depending on model from high to almost zero | Maximal: Model language, file structure, mnemonics; Minimal: zero |
| First GUI set-up | | Negative, not needed without GUI | |
| GUI extension | | Negative, not needed without GUI | |

Source: own research

it quite easy to organize trainings with regard to result analysis.

The yearly 2-3 days CAPRI training sessions from the last decade provide ample observations for that use case, as the exploitation tools of the proprietary CAPRI GUI are identical to GGIG. Generally, users learn relatively fast to work with the exploitation tools. Typically, a one to two hour instruction was sufficient to teach the basic functionalities, for example, how to load a set of results from different scenarios, to navigate between different views, to produce a map or table with relative changes, or to export results to clipboard.

These short instructions also covered some content related information, for example, where to find and how to interpret results on prices, market balances, trade, farm activities, or environmental indicators. With that background, users with sufficient domain knowledge, i.e. with regard to agricultural policies and markets while being trained in economic reasoning, were able to analyse complex scenarios with the exploitation tools without having any technical knowledge about CAPRI (mnemonics, code structure etc.). The participants mentioned these points also regularly in feedback rounds. They were quite diverse with regard to their pre-existing knowledge on economic modelling or software use. The same observations were made in block courses with PhD students.

Over the years, some desk officers in the EU administration received a 2-3 hour introduction of how to exploit CAPRI results based on its GUI and afterwards analyzed results without any experienced user being available. That situation is different from training sessions or a block course where participants can both support each other and draw on the expertise of the trainers. The experiences of these users thus underline further that learning time for that type of GUI seems to be limited. The positive experiences from the

training session, classroom use and individual applications underline that GUIs can indeed help to improve accessibility to results of economic model.

The *second use case* is *running the model*, i.e. to perform a policy or market scenario.[10] The standard case is that users are at first trained in analysing results. Having mastered successfully that step clearly increases the motivation to learn more about the model. In order to perform a policy scenario, the user needs to map a change in legislation into a quantitative change in exogenous parameters. That requires an understanding of the legislation, of the model's methodology and finally, about how to code the change. GGIG can clearly only ease the last bit. To what extent that is possible depends largely on the complexity of both the policy presentation in a model and the scenario. Besides defining controls which change policy related exogenous parameters, GGIG comprises a "scenario definition tool" where pre-existing GAMS code snippets can be changed in an editor and combined to a scenario. The same observations also hold for scenarios that change other exogenous parameters such as macro-variables.

The models mentioned in Table 2 differ considerably in their approaches to define a model run. The two extremes will be presented briefly in the following. In the single farm model FARMDYN, a larger part of the ~170 controls mentioned in table 2 allows to define a model run both with regards to assets of the farm (land, labour, stable, machinery ...), the shocks (mainly prices for output and inputs) and model properties (time horizon, methodological features). A direct editing of changes in the GAMS code is not necessary. For CAPRI, however, the opposite is typical: a user defines a scenario in the GAMS code directly, potentially supported by the scenario definition tool. Existing scenarios are stored in sub-directories and can be re-used. The CAPRI GUI however allows to switch certain modules respectively model features on or off, and to select the years, regions and spatial resolution of the model run.

In opposite to the first use case, the contribution of a GUI to reduced learning costs is therefore far more depending on the model. Especially partial equilibrium models often depict policy instruments close to the law book; CAPRI, to give an example, distinguishes between ~60 different subsidies schemes at farm level. Defining controls at that detail is a tremendous task with probably limited returns, as GAMS coding gives far more flexibility e.g. to define groups of subsidies and to apply changes at group level.

CAPRI provides some further observations for the second use case of running the model; not astonishing, they are fewer than those for the first one. By now, more and more institutions not previously involved in the development of CAPRI apply the model. In most cases, on the job training is not a viable option, as it requires prolonged staff exchange. Hence, a small group of newcomers (or less experienced users) attended short block courses of 1-3 days, not only to learn how to exploit and analyse results as in the yearly training sessions, but also to perform more complex tasks such as to define and run policy scenarios or to construct an ex-ante baseline. The GUI was typically assessed as the easiest part to learn, and the contrast in learning time and success rate is striking between tasks which require own coding efforts and tasks which can be performed by solely using the GUI. A similar experience provides a recent course where a group of master students used the GTAP in GAMS code from Tom Rutherford (RUTHERFORD and HARBOR, 2005) successfully in conjunction with GGIG to develop and analyse their own scenarios, mostly without any support by an experienced user.

Given the number of training sessions and courses, these observations relate to more than one hundred persons, with quite different pre-existing knowledge and talents. The observations differ considerable from those made in early CAPRI training sessions using earlier versions of CAPRI with no or a far less well developed GUI. In these earlier versions, users had to edit the GAMS code and to use GAMS tools to inspect results. Users typically achieved far less, while the trainers permanently had to support participants in overcoming problems related to GAMS coding or assessing results. One might hence summarize that, at least based on the experiences with CAPRI and GGIG, GUIs are indeed able to considerably reduce learning time and steering errors with regard to economic model applications.

The *last use case* relates to the *implementation of a GUI in GGIG*. For the smaller projects reported in Table 1, a trained coder could set up a GUI and a set of exploitation tables in a few hours. In opposite to the smaller- to medium-scale examples mentioned above,

[10] From a technical viewpoint, a model runs means that GGIG passes the status of the control to a GAMS (or R) program. That means that GGIG can also be used to run GAMS programs which, for example, build up the model database. We will, however, not analyse that type of application in here. Firstly, typically a few, quite experienced people are typically involved in these tasks, and secondly, it is far less standardized across models.

CAPRI has a much more complex GUI. Setting up a first working version took considerably longer, also as its implementation led to improvements to GGIG itself and thus was mixed with code development in the GUI generator itself. Equally, there were some legacy questions as the GAMS code needed to work over a longer testing period both under the old and new GUI. From the other examples, the river basin model was the only one where the GUI was added after the main development phase, and perhaps not astonishing, it went along with some refactoring of the GAMS code. The experiences with CAPRI and the river basin model might serve as an indication that it is generally best to develop the GAMS code from the beginning in parallel to the GUI.

After the first version of the CAPRI GUI was set up in GGIG, GAMS coders involved in CAPRI projects have added new features to the projects' GUIs. The same holds for researchers contributing to some of the other projects mentioned above. These experiences seem to underline that at least once a starting implementation based on GGIG is given, the learning time required to expand a GUI seems quite low.

To summarize and conclude, the use cases seem to underline that it is advantageous to complement economic models with a GUI, at least if their use beyond the core group of original developers is required. The largest reduction in learning costs with a GUI can be achieved if users solely exploit existing scenarios. The experiences in that respect seem to indicate that the learning time can be so low that result sets of complex economic models can be used in the classroom or to let informed clients access results. It clearly motivates potential model users if they are able to analyse model runs based on their knowledge with regard to markets, policies, economic theory and economic model methodology without having first to learn a lot of technical detail. Based on that experience about what the model is able to deliver, they can take an informed decision if they want to acquire the necessary skills to master more complex tasks, such as to define and run policy scenarios. For clients, having easy access to results also reduces the black-box character of complex models and might build trust (as long as results make sense). More specifically, the experiences with GGIG so far might also serve as an indication that users seem to learn quickly to use GGIG based GUIs, while the efforts needed to build a GUI in GGIG are limited.

# 6 Summary and Conclusions

Result analysis of economic models and more so model runs require considerably learning efforts as users need to familiarize themselves with the modelling language used, mnemonics and further technical detail. Graphical User Interfaces (GUI) have the potential to not only reduce learning costs, but also to more efficiently steer models and exploit results.

However, developing a proprietary GUI for an economic model is typically a costly task which requires knowledge in software engineering. With GGIG (GAMS Graphical Interface Generator), a relatively simple GUI builder XML for GAMS and R projects is now available where components are defined in an XML file. GUIs generated with GGIG carry user operable controls of which the settings are passed via an include file to GAMS. They allow spawning GAMS processes and merging the results from different GAMS runs for exploitation, supported by a rather flexible reporting tool. GGIG also comprises a set of further utilities, e.g. for SVN updates or to build a documentation of GAMS symbols and files in HTML pages, similar to javadoc. First applications underline that GAMS coders without formal software training are able to build their own GUI, whereas users learn quickly to use the GUI for result analysis and model runs.

# References

ABRAN, A., A. KHELIFI, W. SURYN and A. SEFFAH (2003): Usability meanings and interpretations in ISO standards. In: Software Quality Journal 11 (4): 325-338.

ANG, C. (2004): Optimized recovery of damaged electrical power grids. Master Thesis. Naval Postgraduate School, Monterey, California. In: DOI:http://worldcat.org/oclc/66268376.

BANSE, M., H. GRETHE and S. NOLTE (2005): European Simulation Model (ESIM) in the General Algebraic Modeling System (GAMS): Model Documentation. Humboldt University of Berlin, Universities Hohenheim and Göttingen.

BAZZANI, G. (2005): An integrated decision support system for irrigation and water policy design: DSIRR. In: Environmental Modelling and Software 20 (2): 153-163.

BOUMA, F., B. ELBERSEN, J. ROOS-KLEIN LANKHORST and I. STARITSKY (2010): Deliverable 5.6: Technical Description of Final CCAT Tool, LEI The Hague. In: http://www.wageningenur.nl/upload_mm/a/1/1/6801a81c-7fde-4f46-a244-9a2fc5894250_D5_6TechnicalDescriptionCCATFinalToolFINAL1.pdf.

BRITZ, W. (2012): RegCgeEU+ in GAMS, documentation including the Graphical User Interface, CAPPRI-RD Deliverable 3.2.4. In: http://www.ilr1.uni-bonn.de/agpo/rsrch/capri-rd/docs/d3.2.4.pdf.

– (1999): IT - An Unimportant Ingredient of Large Scale Models? In: Agrarwirtschaft 48-(3/4): 159-162.

– (2011): The Graphical User Interface for CAPRI version 2011. University Bonn, Institute for Food and Resource Economics. In: http://www.capri-model.org/docs/Gui2011.pdf.

– (2014a): GGIG Graphical Interface Generator Programming Guide. University Bonn, Institute for Food and Resource Economics. In: http://www.ilr.uni-bonn.de/agpo/staff/britz/GGIG_programming_guide.pdf.

– (2014b): GGIG Graphical Interface Generator User Guide. University Bonn, Institute for Food and Resource Economics. In: http://www.ilr.uni-bonn.de/agpo/staff/britz/GGIG_user_Guide.pdf.

– (2014c): The Graphical User Interface for CAPRI version 2014. University Bonn, Institute for Food and Resource Economics. In: http://www.capri-model.org/docs/Gui2014.pdf.

BRITZ, W. and J. KALLRATH (2012): Economic Simulation Models in Agricultural Economics: The Current and Possible Future Role of Algebraic Modelling Languages. In: Kallrath, J. (ed.): Algebraic Modelling Systems: Modelling and Solving Real World Optimization Problems. Springer, Heidelberg, Germany: 199-212.

BRITZ, W. and A. LEIP (2009): Development of marginal emission factors for N losses from agricultural soils with the DNDC–CAPRI meta-model. In: Agriculture, Ecosystems & Environment 133 (3-4): 267-279.

BRITZ, W., P. PEREZ DOMINGUEZ and G.P. NARAYANAN (2013): Analyzing results from agricultural large-scale Economic Simulation Model: State-of-the-art and Way Ahead. In: submitted to German Journal of Agricultural Economics.

BRITZ, W. and P. WITZKE (2012): CAPRI model documentation 2012. University Bonn, Institute for Food and Resource Economics. In: http://www.capri-model.org/docs/capri_documentation.pdf.

Brooke, A., D. Kendrick and A. Meeraus (1988): GAMS: A User's Guide. The Scientific Press, Redwood City, California.

BULATEWICZ, T., A. ALLEN, J.M. PETERSON, S. STAGGENBORG, S.M. WELCH and D.R. STEWARD (2013): The Simple Script Wrapper for OpenMI: Enabling interdisciplinary modeling studies. In: Environmental Modelling & Software 39: 283-294.

DE WOLF, D. and Y. SMEERS (2000): The Gas Transmission Problem Solved by and Extension of the Simplex Algorithm. In: Management Science 46 (11): 1454-1465.

DOL, W. and F. BOUMA (2006): The GSE philosophy: a concept of model building as a team activity. LEI-Wageningen UR, The Hague.

DOL, W. (2006): GAMS Simulation Environment. LEI The Hague. In: http://www3.lei.wur.nl/gamstools/gse.doc.

EU (2009): Impact Assessment Guidelines, SEC(2009) 92.

HARRISON, W.J. and K.R. PEARSON (1996): Computing Solutions for Large General Equilibrium Models Using GEMPACK. In: Computational Economics 9 (2): 83-127.

HAVLÍK, P., H. VALIN, A. MOSNIER, M. OBERSTEINER, J.S. BAKER, M. HERRERO, M.C. RUFINO and E. SCHMID (2013): Crop Productivity and the Global Livestock Sector: Implications for Land Use Change and Greenhouse Gas Emissions. In: American Journal of Agricultural Economics 95 (2): 442-448.

HEIDECKE, C. and T. HECKELEI (2010): Impacts of changing water inflow distributions on irrigation and farm income along the Drâa River in Morocco. In: Agricultural Economics 41 (2): 135-149.

HERTEL, T.W. (ed.) (1997): Global Trade Analysis: Modelling and Applications. Cambridge, University Press.

IHAKA, R. and R. GENTLEMAN (1996): R: a language for data analysis and graphics. In: Journal of computational and graphical statistics 5 (3): 299-314.

ISO 9241 (1992/2001): Ergonomics requirements for office with visual display terminals (VDTs). International Organization for Standardization, Geneva.

JANSSEN, S., K. LOUHICHI, A. KANELLOPOULOS, P. ZANDER, G. FLICHMANN, H. HENGSDIJK, E. MEUTER, E. ANDERSEN, H. BELHOUCHETTE, M. BLANCO, N. BORKOWSKI, T. HECKELEI, M. HECKER, H. LI, A. OUDE LANSINK, G. STOKSTAD, P. THORNE, H. VAN KEULEN and M. VAN ITTERSUM (2010): A Generic Bio-Economic Farm Model for Environmental and Economic Assessment of Agricultural Systems. In: Environmental Management 46 (6): 862-877.

KALLRATH, J. (2012): Algebraic Modeling Languages: Introduction and Overview. In: Kallrath, J. (ed.): Algebraic Modelling Systems: Modelling and Solving Real World Optimization Problems. Springer, Heidelberg, Germany: 3-10.

KNAPEN, M.J.R., S.J.C. JANSSEN, O.R. ROOSENSCHOON, P.J.F.M. VERWEIJ, W. DE WINTER, M. UITERWIJK and J.E. WIEN (2013): Evaluating OpenMI as a model integration platform across disciplines. In: Modelling & Software 39: 274-282.

KUHN, A., V. MULINDABIGWI, M. JANSSENS, G. STEUP, T. GAISER, H. GOLDBACH, I. GRUBER and E. GANDONOU (2010): Impacts of Global Change on food security in Benin. In: Speth, P., M. Christoph and B. Dieckkrüger (eds.): Impacts of Global Change in the Hydrological Cycle in West and Northwest Africa. Springer, Berlin: 454-483.

KUHN, A., P. VAN OEL and F. MEINS (2012): The Lake Naivasha Hydro-Economic Basin Model (LANA-HEBAMO) - A Technical Documentation. DFG Research Unit 1501, Sub-Project B2, Technical Paper 10/2012. Universität Bonn.

LANIAK, G.F., G. OLCHIN, J. GOODALL, A. VOINOV, M. HILL, P. GLYNN, G. WHELAN, G. GELLER, N. QUINN, M. BLIND, S. BECKHAM, S. REANEY, N. GABER, R. KENNEDY and A. HUGHES (2013b): Integrated Environmental Modeling: A Vision and Roadmap for the Future. In: Environmental Modelling & Software 39: 3-23.

LANIAK, G.F., A.E. RIZZOLI and A. VOINOV (2013a): Thematic Issue on the Future of Integrated Modeling Science and Technology. In: Environmental Modelling & Software 39: 1-2.

LEIP, A., G. MARCHI, R. KÖBLE, M. KEMPEN, W. BRITZ and C.C. LI (2008): Linking an economic model for European agriculture with a mechanistic model to estimate ni-

trogen losses from cropland soil in Europe. In: Biogeosciences 5 (1): 73-94.

LENGERS, B. and W. BRITZ (2012): The choice of emission indicators in environmental policy design: an analysis of GHG abatement in different dairy farms based on a bioeconomic model approach. In: Review of Agricultural and Environmental Studies 93 (2): 117-144.

M`BAREK, R., W. BRITZ, A. BURRELL and J. DELINCE (2012): An integrated Modelling Platform for Agro-economic Commodity and Policy Analysis (iMAP). JRC Scientific and Policy Reports, Luxembourg. Publications Office of the European Union, Luxembourg, 25267 EN.

MCDONALD, S., K. THIERFELDER and T. WALMSLEY (2012): Globe v2: A SAM Based Global CGE Model using GTAP Data. Oxford Brookes University, Departement of Economics, Oxford, UK. In: http://www.cgemod. org. uk/globev2_2012.pdf.

MOSS, J.E., J.C. BINFIELD, L. ZHANG, M. PATTON and I.S. KIM (2011): A Stochastic Analysis of the Impact of Volatile World Agricultural Prices on European and UK Agriculture. In 85th Annual Conference of the Agricultural Economics Society, Warwick University: 18-20.

MORARI, M. and I.E. GROSSMANN (eds.) (1991): Chemical Engineering Optimization Models with GAMS. CACHE Process Design Case Studies 6. CACHE Austin, Texas.

OECD (2007): Documentation of the AGLINK-COSIMO Model. Organization for Economic Co-operation and Development, Paris. In: http://ww.oecd.org/officialdocuments/publicdislaydocumentpdf/cote=AGR/CA/APM(2006)16/FINAL&docLanguage=En.

PEARSON, K., M. HORRIDGE and A.N. PRATT (2003): Hands-on Computing with RunGTAP and WinGEM to Introduce GTAP and GEMPACK. The Center for Global Trade Analysis. In: https://www.gtap.agecon.purdue. edu/resources/download/2692.pdf.

PODHORA, A., K. HELMING, L. ADENÄUER, T. HECKELEI, P. KAUTTO, P. REIDSMA, K. RENNINGS, J. TURNPENNY and J. JANSEN (2013): The policy-relevancy of impact assessment tools: Evaluating nine years of European research funding. In: Environmental Science and Policy 31: 85-95.

RAILSBACK, S.F., S.L. LYTINEN and S.K. JACKSON (2006): Agent-based simulation platforms: review and development recommendations. In: Simulation 82 (9): 609-623.

REBENNACK, S., J. KALLRATH and P.M. PARDALOS (2010): Energy Portfolio Optimization for Electric Utilities: Case Study for Germany. In: Bjorndal, E., M. Bjorndal, P.M. Pardalos and M. Ronnqvist (eds.): Energy, Natural Resources and Environmental Economics. Springer, Berlin, Heidelber: 221-246.

RICHMOND, B. and S. PETERSON (1997): An introduction to systems thinking. High Performance Systems, Hanover, New Hampshire.

RIZZOLI, A.E., J.J.F. WIEN, R. KNAPEN, L. RUINELLI, I. ATHANASIADIS and B. JONSSON (2009): Updated version of final design and of the architecture of SEAMLESS-IF Report No.47, SEAMLESS integrated project, EU 6th Framework Programme, contract no. 010036-2. In: http://www.SEAMLESS-IP.org.

ROBINSON, M. and P. VOROBIEV (2003): Swing. Second Edition. Manning, Greenwich, Connecticut.

RUTHERFORD, T. and A. HARBOR (2005): GTAP6inGAMS: The Dataset and Static Model. Prepared for the Workshop "Applied General Equilibrium Modeling for Trade Policy Analysis in Russia and the CIS". The World Bank Resident Mission, Moscow, December 1-9, 2005. In: http://www.mpsge.org/gtap6/gtap6gams.pdf.

SALAMON, P., F. CHANTREUIL, T. DONNELLAN, E. ERJAVEC, R. ESPOSTI, K.F. HANRAHAN, M. VAN LEEUWEN, F. BOUMA, W. DOL and G. SALPUTRA (2008): How to deal with the challenges of linking a large number of individual national models: the case of the AGMEMOD Partnership. In: German Journal of Agricultural Economics 57 (8): 373-378.

VAN ITTERSUM, M., F. EWERT, T. HECKELEI, J. WERY, J. ALKAN OLSSON, E. ANDERSEN, I. BEZLEPKINA, F. BROUWER, M. DONATELLI, G. FLICHMANN, L. OLSSON, A.E. RIZZOLI, T. VAN DER WAL, J.E. WIEN and J. WOLF (2008): Integrated assessment of agricultural systems – A component-based framework for the European Union (SEAMLESS). In: Agricultural Systems 96 (1-3): 150-165.

VAN LEEUWEN, M., F. BOUMA, F. CHANTREUIL, W. DOL, E. ERJAVEC, K.F. HANRAHAN, P. SALAMON and G. SALPUTRA (2012): AGMEMOD Model. In: The Future of EU Agricultural Markets by AGMEMOD. Springer, Dordrecht, Netherlands: 45-74.

VAN MEIJL, H. and G. WOLTJER (2012): The development of the MAGNET strategy. Paper presented at the 15th Annual Conference on Global Economic Analysis, Geneva, Switzerland. In: https://www.gtap.agecon.purdue. edu/resources/res_display.asp?RecordID=3956.

VERWEIJ, P.J.F.M., M.J.R. KNAPEN, W.P. DE WINTER, J.J.F. WIEN, J.A. TE ROLLER, S. SIEBER and J.M.L. JANSEN (2010): An IT perspective on integrative environmental modelling: the SIAT case. In: Ecological Modelling 221 (18): 2167-2176.

WIECK, C., S.W. SCHLÜTER and W. BRITZ (2012): Assessment of the Impact of Avian Influenza Related Regulatory Policies on Poultry Meat Trade and Welfare. In: The World Economy 35 (8): 1037-1052.

WITTEN, I.H.E. FRANK and M.A. HALL (2011): Data Mining Practical Machine Learning Tools and Techniques. Third edition. Elsevier, Amsterdam.

WOLTJER, G. (2013): Simplifying general equilibrium analysis through a modular structure: MAGNET. Paper for the 16th Annual Conference on Global Economic Analysis "New Challenges for Global Trade in a Rapidly Changing World", June 12-14 2013, Shanghai, China. In: https://www.gtap.agecon.purdue.edu/resources/res_display.asp?RecordID=4097.

**DR. WOLFGANG BRITZ**
Universität Bonn, Institut für Lebensmittel- und Ressourcenökonomik
Nussallee 21, 53229 Bonn
e-mail: wolfgang.britz@ilr.uni-bonn.de